Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Rok Ritlop

Umetno življenje na primeru kolonije inteligentnih agentov

diplomsko delo
univerzitetni študij

Mentor: izr. prof. dr. Marko Robnik-Šikonja

Ljubljana 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Področje umetnega življenja z računalniškimi modeli simulira in proučuje različne vidike življenja. Predvsem nas zanimajo pojavi, ki so sicer značilni za življenjske oblike, npr. medsebojno komuniciranje, sodelovanje, tekmovanje in mehanizmi razmnoževanja. Zasnujte in izvedite model preprostega umetnega okolja, v katerem poteka evolucija inteligentnih agentov, ki so predstavljeni z nevronskimi mrežami, vendar uporabite tudi ideje z drugih področij evolucijskega računanja. Razvito okolje uporabite za simulacijo življenja, ki ga grafično vizualizirajte. Simulirajte in analizirajte nekaj scenarijev življenja in rezultate statistično ovrednotite.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Rok Ritlop, z vpisno številko **63080110**, sem avtor diplomskega dela z naslovom:

Umetno življenje na primeru kolonije inteligentnih agentov.

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Marka Robnika-Šikonje,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

Ljubljana, 29. 6. 2015 Podpis avtorja:

Zahvaljujem se mentorju, izr. prof. dr. Marku Robniku-Šikonji, za strokovno pomoč in podporo pri izdelavi diplomske naloge.

Mami, babici in dediju se zahvaljujem za podporo in vzpodbude pri študiju, mojemu dekletu Maji pa za podporo in ideje pri izdelavi diplomske naloge.

Za konec bi se zahvalil kolegom Pavlu, Roku in Martinu; z vami je bil študij dosti lažji in zabavnejši.

Kazalo

Povzetek

Izdelali smo simulacijo umetnega življenja na primeru kolonije inteligentnih posameznikov (agentov), pri čemer smo uporabili kombinacijo nevronskih mrež, genetskih algoritmov in optimizacije s kolonijo mravelj. Vsak agent ima svoje možgane, implementirane z nevronsko mrežo. Večje število agentov tvori kolonijo, ki ima interakcijo z okoljem preko nabiranja in shranjevanja hrane, med seboj pa komunicirajo s pomočjo feromonskih sledi. Preko dedovanja in mutacije že obstoječih nevronskih mrež je omogočen razvoj kolonije. S simulacijo smo pridobili podatke o učinkovitosti kolonij z različnimi stopnjami mutacije (0,01 do 0,10) in rezultate med seboj primerjali. Kolonije z nizkimi stopnjami mutacije so bile v splošnem manj uspešne, kolonije z visokimi stopnjami mutacije pa so uspešno in hitro razvile le nekatera vedenja. Za najbolj uspešne so se izkazale kolonije s stopnjo mutacije 0,05, ki predstavlja dobro ravnovesje med kreiranjem novih agentov in kopiranjem že obstoječih uspešnih agentov. Simulacija omogoča spreminjanje parametrov in predstavlja dobro osnovo za nadaljnji razvoj in nadgradnje ter dodajanje bolj kompleksnih vedenj agentov.

Ključne besede: umetno življenje, nevronske mreže, genetski algoritmi, optimizacija s kolonijo mravelj.

Abstract

We developed a simulation of artificial life with a colony of intelligent individuals (agents) by using a combination of neural networks, genetic algorithms, and ant colony optimization. Each agent has its own brain implemented as a neural network. Several agents form a colony which interacts with the environment by gathering and storing food. They communicate using pheromone trails. Through the processes of inheritance and mutation of agents' brain the colony can develop continuously. With simulation we gathered the information on the effectiveness of colonies with varying rates of mutation (from 0.01 to 0.10) and compared the results. The colonies with low mutation rates were overall less successful, while the colonies with high mutation rates were successful in developing only certain behaviours. The most successful colonies used the mutation rate of 0.05, which presents a good balance between creating new agents and copying the existing successful agents. The simulation allows adjustment of parameters and presents a good basis for further development and adding more complex agent behaviours.

Keywords: artificial life, neural networks, genetic algorithms, ant colony optimization

1. Uvod

Ljudje že od nekdaj stremimo k razumevanju svojega okolja in sebe. Ustvarjamo si tako laične predstave o dogajanju okoli nas kot uporabljamo znanstvene pristope, da bi razširili svoje znanje. Z razvojem tehnologije nam pri tem vse bolj pomaga tudi računalništvo, ki nam med drugim omogoča, da lahko s pomočjo različnih simulacij preverjamo, ali naša dosedanja spoznanja o delovanju živih organizmov in človeškem vedenju držijo. Na podlagi tega, kar vemo o našem lastnem delovanju, predvsem pa inteligentnem vedenju (kot sta reševanje problemov in načrtovanje), se je razvilo področje umetne inteligence, ki se skuša preko simulacije čim bolj približati dejanskemu inteligentnemu vedenju živih bitij. Na drugi strani pa si želimo preizkušati, kakšne vse oblike vedenja in življenja lahko nastanejo na podlagi bolj osnovnih vedenj. Zanima nas, ali bodo umetne oblike življenja razvile vedenja, kot smo jih pričakovali ali pa bodo razvile čisto nova, drugačna, a vseeno na okoliščine prilagojena in uspešna vedenja. Tovrsten pristop, imenovan umetno življenje, nam omogoča vpogled v evolucijo in razvoj življenja, hkrati pa nam nudi mnoge nove načine reševanja različnih problemov, tudi takih, ki jih je težko rešiti s klasičnimi pristopi.

Želeli smo izdelati simulacijo umetnega življenja na primeru kolonije inteligentnih posameznikov (v nadaljevanju agenti). Pri tem smo se odločili uporabiti posamezne elemente iz več različnih načinov reševanja problemov, in sicer nevronskih mrež, genetskih oz. evolucijskih algoritmov in optimizacije

s kolonijo mravelj (*ant colony optimization*).

Diplomsko delo je razdeljeno na šest poglavij. V prvem poglavju bomo predstavili nekaj osnovnih konceptov, ki smo jih deloma uporabili za izdelavo simulacije. V drugem poglavju bomo opisali simulacijo z vsebinskega vidika in predstavili posamezne dele simulacije (okolje z viri hrane in domom ter agente, ki tvorijo kolonijo). V tretjem poglavju bomo podali tehnični opis programa, in sicer tehnični opis delovanja grafičnega okna z uporabniškim vmesnikom, implementacijo vedenja (premikanje, nabiranje hrane, hranjenje itd.) in generiranja novih agentov (dedovanje in mutacija), način komunikacije med agenti (zapis in prepoznavanje feromonov), določanje učinkovitosti agentov (točkovanje) in način beleženja podatkov. V četrtem poglavju bomo predstavili nekaj zanimivih ugotovitev o tipičnih vzorcih vedenja posameznih entitet in kolonije kot celote. V petem poglavju bomo podali nekaj idej za izboljšanje in razširitev obstoječe simulacije. Šesto poglavje predstavlja sklepe in ugotovitve.

1.1 Umetna inteligenca in umetno življenje

Pri preučevanju in razumevanju življenja ter inteligentnih oblik vedenja sta se razvili dve veji, in sicer umetna inteligenca (*artifical intelligence*; v nadaljevanju UI) in umetno življenje (*artifical life*; v nadaljevanju UŽ). UI se ukvarja z razvojem računalniških sistemov, ki so sposobni izvajati naloge, ki običajno zahtevajo inteligenco ljudi. Vključuje modeliranje tako vsakodnevnega kot ekspertnega znanja in mišljenja pri ljudeh, reševanje problemov, načrtovanje itd. Tipične naloge so diagnosticiranje bolezni, razumevanje zgodb, odgovarjanje na vprašanja in igranje šaha. UI uporablja pristop od zgoraj navzdol. Za ta pristop je značilno, da opazovani sistem (npr. vedenje) razgrajujemo na podsisteme, dokler ne pridemo do osnovnih gradnikov (npr. posameznih vzorcev vedenja). Na drugi strani uporablja UŽ pristop od spodaj navzgor,

za katerega je značilno sestavljanje osnovnih gradnikov sistemov, da bi dobili bolj kompleksen sistem. Glavni cilj je dobiti vpogled v življenje, kot je in v življenje, kot bi lahko bilo. Medtem ko se UI osredotoča na modeliranje kognitivnih nalog, ki večinoma vključujejo le posameznike, je UŽ vezano na biološko perspektivo preučevanja inteligentnega vedenja, ki poudarja evolucijo in razvoj kognicije preko interakcije in skozi več generacij. Osnova UŽ so različni računalniški modeli, ki se uporabljajo tudi v biologiji in ekoloških študijah (npr. teorija iger in tehnike optimizacije). UŽ tem tehnikam doda simulacijo umetnega živčnega sistema. Ta omogoča agentom premikanje v umetnem okolju, ki ima strukturo (običajno dvodimenzionalnega) prostora. Osnovna enota manipulacije je populacija, saj imajo agenti omejeno življenjsko dobo, namesto njih pa se rojevajo novi. Genom se spreminja preko genetskih ali preko adaptivnih učnih algoritmov. Dolgoročni cilj UŽ je pridobiti vpogled v evolucijo in naravo človeške inteligentnosti skozi modeliranje evolucije komunikacije in sodelovalnega vedenja pri nižjih življenjskih oblikah. Tipične naloge UŽ vključujejo izogibanje plenilcem, nabiranje hrane in iskanje partnerja [1].

Konstrukcija tovrstnih modelov poteka v več fazah – identifikacija fenomena, ki ga želimo preučiti, konstruiranje umetnega sistema, ki vzpostavlja interakcijo z okoljem, beleženje preučevanega fenomena in implementacija popravkov glede na razlike med vedenjem sistema in preučevanim fenomenom. Pri preučevanju inteligentnega vedenja so v ospredju teme, ki jih običajno preučujejo tudi druge vede (npr. etologija in ekologija v primeru živali ter psihologija in sociologija v primeru ljudi). Glavni poudarek je na ugotavljanju, kaj dela vedenje inteligentno in adaptivno ter kako se pojavi. Vedenje je definirano kot pogosto opažena interakcija med lastnostmi in procesi sistema ter lastnostmi in procesi okolja. Vedenje sistema je inteligentno, če maksimizira možnost za samoohranitev sistema v določenem okolju. Pri preučevanju nas ne zanima samo fizično vedenje, ampak principi, ki jih lahko opazimo na nivoju vedenja. Primer tovrstnega pristopa je preučevanje formiranja poti pri mravljah s pomočjo sklopa specifičnih vedenjskih pravil [2].

1.2 Nevronske mreže

Ideja o uporabi nevronskih mrež v računalništvu sega že v 40. leta prejšnjega stoletja. Začetke tega področja računalništva in raziskovanja pripisujejo članku avtorjev McCullock in Pitts, ki je izšel leta 1943 [3]. Ob iskanju možnih izboljšav v računalništvu, so znanstveniki skušali uporabiti tudi znanja iz nevrobiologije in so tako razvili sistem, ki se zgleduje po delovanju možganov živih bitij. Sistem je sestavljen iz majhnih in preprostih računskih enot (imenovanih tudi nevroni), ki so med seboj povezane tako, da tvorijo mrežo. Posamezni nevroni so preproste enote, ki pa lahko tvorijo kompleksne sisteme [4]. Nevronske mreže so postale uporabne, ko so razvili načine za njihovo učenje, kar omogoča, da posamezno mrežo prilagodimo za reševanje poljubnih problemov [3].

Nevronska mreža je v splošnem sestavljena iz petih komponent [5]:

1. usmerjenega grafa;
2. spremenljivk, ki predstavljajo stanja vozlišč v grafu (t. i. nevron);
3. uteži za vsako povezavo v grafu;
4. odmika za vsako vozlišče (*bias*) in
5. prenosne funkcije (*transfer function*) za vsako vozlišče, ki določa stanje vsakega od vozlišč kot funkcijo odmika, uteži na vhodnih povezavah in stanja z njim povezanih vozlišč.

Stanje posameznega nevrona k (y_k) tako v splošnem izračunamo s prenosno funkcijo (φ), v kateri predstavljajo spremenljivke x_0 do x_m vrednosti vhodnih nevronov, spremenljivke ω_0 do ω_m pa uteži na povezavah. Ena izmed možnih implementacij odmika (*bias*) je nastavitev vrednosti x_0 na 1, tako da je $x_{k0} = \omega_{k0}$ (Enačba 1.1).

$$y_k = \varphi \sum_{j=0}^{m} \omega_{kj} x_j \tag{1.1}$$

Struktura mreže določa, kateri nevron lahko vpliva na nek drug nevron, uteži pa določajo moč vpliva [4].

Nevronska mreža je lahko enosmerna (*feed-forward network*), kar pomeni, da nima ciklov. Vhodni nevroni so tisti, do katerih ne vodi nobena povezava, izhodni nevroni so tisti, ki nimajo nobene izhodne povezave, ostali nevroni pa so skriti. Ko so poznane vrednosti začetnih oz. vhodnih nevronov, se lahko na podlagi povezav in njihovih uteži nastavijo vrednosti vseh ostalih nevronov. Tovrstna mreža iz vhodnih podatkov na ta način izračuna izhodne vrednosti. Sestavljena je lahko iz večjega števila nivojev (t.i. večnivojska enosmerna nevronska mreža; *layered feed-forward network*), pri čemer velja, da je vsaka pot od vhodnih do izhodnih nevronov enako dolga, n-ti nivo mreže pa vsebuje vsa vozlišča, do katerih se pride v n korakih. Mreža je polno povezana, če je vsako vozlišče na nivoju n povezano z vsemi vozlišči nivoja $n+1$ za vse nivoje n. Prednost opisanih mrež je, da omogočajo veliko mero posploševanja, saj velikokrat pravilno izračunajo izhodne vrednosti tudi na podatkih, ki niso bili vključeni v učno množico. S pomočjo učnih algoritmov, npr. vzvratnega razširjanja napake (*back-propagation*) lahko pogosto najdemo dober nabor uteži [5].

Za izračun dobrega približka zveznih funkcij je dovolj večnivojska nevronska mreža z le enim skritim nivojem (teorem univerzalne aproksimacije za nevronske mreže – *universal approximation theorem*) [6]. Optimalno število nevronov v skritem nivoju je odvisno tako od arhitekture nevronske mreže kot od problema samega, vendar se lahko v grobem odločimo za vrednost med številom vhodnih in izhodnih nevronov [7].

Ker se nevronske mreže lahko učijo na podlagi realnih podatkov, imajo širok spekter uporabnosti – pri klasifikaciji, prepoznavanju vzorcev (prepoznavanje

obrazov) in zaporedij (govor, prepoznava rokopisa), obdelavi podatkov, filtriranju, kompresiji, kontroli (npr. vozil), postavljanju diagnoz, napovedovanju časovnih vrst, modeliranju itd.

1.3 Genetski algoritmi

Genetske algoritme štejemo med evolucijske algoritme. Ti se od drugih algoritmov v splošnem ločijo po tem, da operirajo s populacijo agentov, kjer vsak agent predstavlja možno rešitev danega problema. Vsakemu od njih je, glede na to, kako dobro rešitev predstavlja, dodeljen rezultat (*fitness score*). Pri uporabi evolucijskih algoritmov se moramo odločiti glede naslednjih treh pomembnih parametrov [4]:

- **Predstavitev osebkov** – evolucijski algoritmi operirajo z 'genetskimi' predstavitvami poskusnih rešitev (ki so večkrat zapisane v obliki zaporedja realnih ali celih števil). Določiti moramo tudi funkcijo, ki predstavitev pretvori v fenotip rešitve.
- **Ocena uspešnosti** – je funkcija, ki oceni vsakega agenta glede na to, kako dobro reši problem.
- **Generiranje novih agentov** – ta del evolucijskih algoritmov predstavljajo operatorji za generiranje novih agentov na podlagi enega (npr. mutacija) ali dveh staršev (npr. križanje genov).

Algoritem se izvaja v več korakih [5]:

- **inicializacija** populacije;
- **ocena uspešnosti** agenta;
- **reprodukcija, ki poteka v dveh korakih:**

- izbran je eden ali več staršev, pri čemer je izbor naključen, vendar odvisen od uspešnosti;
- spreminjanje genetskega zapisa staršev z operatorji (npr. mutacija in križanje genov).

Prednost genetskih algoritmov je, da mnogokrat najdejo ravnotežje med ohranjanjem že obstoječih in generiranjem novih rešitev danega problema. Analize kažejo, da lahko genetski algoritmi mnogokrat hitro najdejo zelo dobro rešitev tudi v veliki in kompleksni množici rešitev. S tem, ko v prostoru rešitev preverjamo več rešitev hkrati, se namreč izogibamo nevarnosti konvergence k lokalnemu minimumu. Na drugi strani pa so genetski algoritmi občutljivi na kontrolne parametre kot so na primer velikost vzorca, obseg rekombinacij in mutacij ter na način selekcije. Tako kot je naravna evolucija počasna, je tudi učenje preko umetnega sistema evolucije počasno, zato uporaba teh algoritmov ni vedno časovno ekonomična. Poleg tega se zastavlja vprašanje, ali lahko rešitve, dobljene na problemih manjšega obsega, posplošimo tudi na obsežnejše probleme [3].

Evolucijske algoritme lahko uporabimo kot pomoč pri izdelavi nevronskih mrež, pri čemer raziskovalci ugotavljajo, da je boljše uporabiti genetske algoritme, ki podpirajo nevronske mreže, kot pa obratno. Ena izmed možnosti za uporabo genetskih algoritmov kot podporo nevronskim mrežam je uporaba algoritma za iskanje najbolj optimalne kombinacije uteži v nevronski mreži. Vsak agent tako predstavlja eno izmed možnih kombinacij uteži v nevronski mreži [3].

1.4 Optimizacija s kolonijo mravelj

Optimizacija s pomočjo kolonije mravelj (*ant system*) predstavlja enega izmed hevrističnih algoritmov za reševanje različnih optimizacijskih problemov.

Algoritem temelji na poznavanju tipičnega vedenja mravelj, pri katerih je zanimivo, da skoraj slepe najdejo najkrajšo pot od mravljišča do vira hrane in nazaj. Raziskovalci so ugotovili, da za to uporabljajo komunikacijo s pomočjo feromonskih sledi. Premikajoča se mravlja spušča feromon v različnih količinah na tla in tako označi pot. Vsaka posamezna mravlja se giba skoraj naključno, če pa naleti na sled druge mravlje se z veliko verjetnostjo odloči, da ji bo sledila, kar še ojača že obstoječo feromonsko sled. Pojavi se pozitivna povratna zanka, saj je verjetnost, da bo mravlja izbrala dano pot, povečana za število mravelj, ki so že prej izbrale to pot. Tako proces ojačuje samega sebe, kar povzroča hitro konvergenco. V optimizacijskem algoritmu predstavlja pot mravlje posamezno rešitev problema [8].

Kot primer tipičnega vedenja mravelj znotraj kolonije lahko navedemo eno izmed vrst mravelj (*Tetramorium Caespitum*), ki so se razvile v okolju, kjer je na voljo velika količina hrane. Ker nimajo dolgoročnega spomina, se pri nabiranju hrane in sprejemanju odločitev zanašajo predvsem na komunikacijo znotraj kolonije. Opazimo lahko tri tipe vedenja [9]:

- **Skupinsko rekrutiranje** (*group-recruitment*) – ko mravlja najde nov vir hrane, se vrne v gnezdo, kjer skuša druge mravlje privabiti, da bi ji sledile do vira hrane. Medtem spušča feromonske sledi.
- **Masovno rekrutiranje** (*mass-recruitment*) – če je vir hrane dovolj velik in je skupinsko rekrutiranje uspešno, to vodi do masovnega rekrutiranja, saj je feromonska sled dovolj ojačana, da ji mravlje sledijo tudi brez vodenja.
- **Naključno raziskovanje** (*random exploration*) – ta tip vedenja se lahko pojavlja v katerikoli fazi iskanja hrane. Mravlja, ki sledi feromonski sledi, se odloči zapustiti sled, da bi raziskala še neodkrita področja in morda našla še več hrane oz. bolj učinkovito pot do obstoječega vira hrane. Verjetnost takšnega dogodka je obratno sorazmerna moči feromonske sledi in premo sorazmerna oddaljenosti od gnezda.

1.5 Pregled sorodnih del

Področje simulacije umetnega življenja, kamor lahko umestimo naše delo, lahko razdelimo glede na način predstavitve osebkov na štiri sklope – predstavitev osebkov s pomočjo programa, z individualnimi moduli, z vnaprej določenimi vedenji in spremenljivimi parametri ter z nevronskimi mrežami. Simulacija, ki smo jo izdelali, sodi v zadnjo kategorijo. Ta predstavlja simulacije, v katerih se osebki učijo in razvijajo z uporabo nevronskih mrež, pri čemer je poudarek pogosto na učenju (in ne toliko na naravni selekciji). Primeri takšnega tipa simulacije so različni projekti in igre, ki jih na kratko predstavljamo:

- **Creatures** – računalniška igra, v kateri igralec vzgaja življenjske oblike po imenu Norn. Uči jih preživeti, raziskovati svet, brani jih pred drugimi oblikami življenja in jih uči jezika.
- **Critterding** – 3D simulator, v katerem se bitja borijo za preživetje z razvojem tako fizičnega telesa (glave z usti in trupa) kot nevronske mreže.
- **3DVCE** – program, namenjen zbiranju podatkov o evoluciji. Podatke zbira prek prostovoljcev, ki simulacijo poženejo, rezultate pa pošljejo nazaj avtorjem. Uporabnik lahko nastavi med drugim fizične omejitve telesa, velikost in čas obstoja populacije, stopnjo mutacije in ostale splošne nastavitve evolucijskih algoritmov. Avtorji so na podlagi zbranih podatkov prišli do zanimivih zaključkov, kot so npr. štirje najbolj pogosti tipi telesa.
- **Polyworld** – dvodimenzionalno okolje, v katerem se nahaja populacija trapezoidnih agentov, ki iščejo hrano, iščejo partnerje, imajo potomce in se spopadajo med seboj. Ker imajo agenti sposobnosti, kot so hranjenje (s trupli in hrano), vid, razmnoževanje in napadanje, je pri kasnejših

stopnjah evolucije možno opaziti mnogo zanimivih tipov obnašanj, kot so na primer kanibalizem, plenilec in plen ter posnemanje [10].

- **Evolucija nevronskih mrež** – diplomsko delo, v katerem avtor s postavitvijo ustreznega modela umetnega življenja in simulacijo organizmov v okolju s hrano predstavi evolucijo nevronskih mrež. S pomočjo učenja organizmi razvijajo nevronsko mrežo, ki jim omogoča učinkovito iskanje hrane in s tem preživetje [11].

2. Opis umetnega sveta

Simulacija, ki smo jo ustvarili, je sestavljena iz okolja, v katerem se nahajajo viri hrane in dom. S tem okoljem imajo interakcijo agenti, ki tvorijo kolonijo. Vsak izmed agentov ima možgane, ki so predstavljeni z nevronsko mrežo. Kolonija stremi k ustvarjanju agentov, ki bodo čim bolje funkcionirali znotraj danega okolja, kar pomeni, da morajo imeti agenti optimalno prilagojene uteži v nevronskih mrežah, ki predstavljajo njihove možgane. Da bi to dosegli, smo uporabili kombinacijo genetskih algoritmov in nevronskih mrež. Genetski algoritem uporabljamo za iskanje optimalne kombinacije uteži v nevronski mreži. Znotraj algoritma smo zagotovili vse tri parametre genetskih algoritmov:

- **predstavitev osebkov** – vsak agent ima lastno nevronsko mrežo z unikatno kombinacijo uteži na povezavah med nevroni in predstavlja eno izmed poskusnih rešitev problema optimalnega nabiranja hrane;

- **ocena uspešnosti** – vsak agent nabira točke do svoje smrti, pri čemer je s točkami nagrajen za vedenja, ki koloniji omogočajo preživetje;

- **generiranje novih agentov** – za generiranje novih agentov uporabljamo dva mehanizma, in sicer mutacijo in naključno generiranje agentov.

2.1 Okolje

Okolje je dvodimenzionalna ravnina, v kateri se nahajajo domovi posameznih kolonij in viri hrane. Ko se ustvari *dom*, se na isti lokaciji ustvari večje število agentov, ki to lokacijo razumejo kot svoj dom, v katerega lahko nosijo hrano ali pa se hranijo iz zalog. Da bi se lahko generacije nadaljevale, se v vsakem trenutku v domu shranjujejo podatki o najbolj uspešnih agentih (tudi če so že umrli). Ko agent umre, se na poziciji doma rodi nov agent, ki je potomec enega izmed najbolj uspešnih agentov, pri čemer je prisotna določena stopnja mutacije.

Viri hrane so lahko neskončni ali končni. Če je vir hrane končen, se lahko namesto izčrpanega vira pojavi na drugi lokaciji nov vir.

2.2 Agenti

Vedenje agentov je nadzorovano preko nevronske mreže oz. možganov. Vsak izmed agentov prejema v vsakem trenutku naslednje vhodne podatke:

- ali je doma (*isAtHomeBin()*),
- ali se nahaja tik ob oz. na hrani (*isAtFoodSourceBin()*),
- ali se nahaja v bližini hrane (*canSeeFoodBin()*),
- kolikšna je stopnja njegove lakote (*hunger()*),
- koliko hrane nosi (*carryLoad()*) in
- ali zaznava feromon (ali se v bližini nahaja feromon; *smellsPheromonesBin()*).

Glede na pridobljene vhodne podatke se vsak izmed agentov odloči za eno izmed možnih vedenj oz. izhodov, ki so razdeljeni v vnaprej določene kategorije, in sicer:

- hojo naokoli (*ROAM*),
- hojo v smeri doma (*GO_HOME*),
- hojo v smeri hrane (pod pogojem, da je hrana v njegovi neposredni bližini; *GO_TO_FOOD*),
- pobiranje hrane (*HARVEST_FOOD*),
- hranjenje s hrano, ki jo agent nosi (*EAT_OWN_FOOD*),
- hranjenje s hrano, ki je shranjena doma in je v skupni uporabi kolonije (*EAT_COLONY_FOOD*),
- odlaganje hrane doma (*STORE_FOOD*) in
- sledenje feromonu (*FOLLOW_PHEROMONES*).

Vsak izmed agentov lahko *umre* zaradi lakote, zaradi neaktivnosti ali zaradi starosti.

2.3 Kolonija

Kolonija je sestavljena iz večjega števila agentov, ki so po osnovni strukturi enaki, vendar ima vsak svoje *možgane*, ki glede na pridobljene vhodne podatke uravnavajo njegovo vedenje. Velikost kolonije je konstantna, saj se *rodi* toliko agentov, kolikor jih *umre*. Agenti med seboj preprosto komunicirajo s pomočjo *feromona*, ki ga puščajo za seboj oz. mu sledijo.

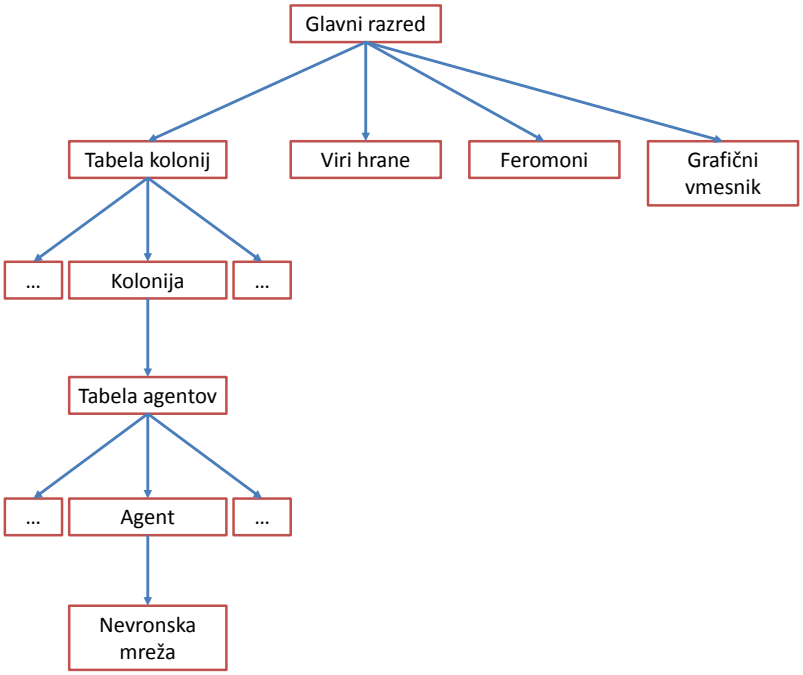
3. Opis simulacijskega programa

Simulacijo smo razvili v programskem jeziku Java, pri čemer smo za grafični prikaz uporabili grafično knjižnico OpenGL.

V nadaljevanju bomo podrobneje opisali arhitekturo aplikacije, izgled in funkcionalnosti grafičnega vmesnika, načine obnašanja agentov in njihovo odzivanje na okolje ter generiranje novih agentov, način komunikacije med agenti s pomočjo feromona, točkovanje oz. oceno uspešnosti agentov, časovnik in beleženje podatkov za obdelavo oz. vodenje dnevnika dogodkov.

3.1 Arhitektura simulacijskega programa

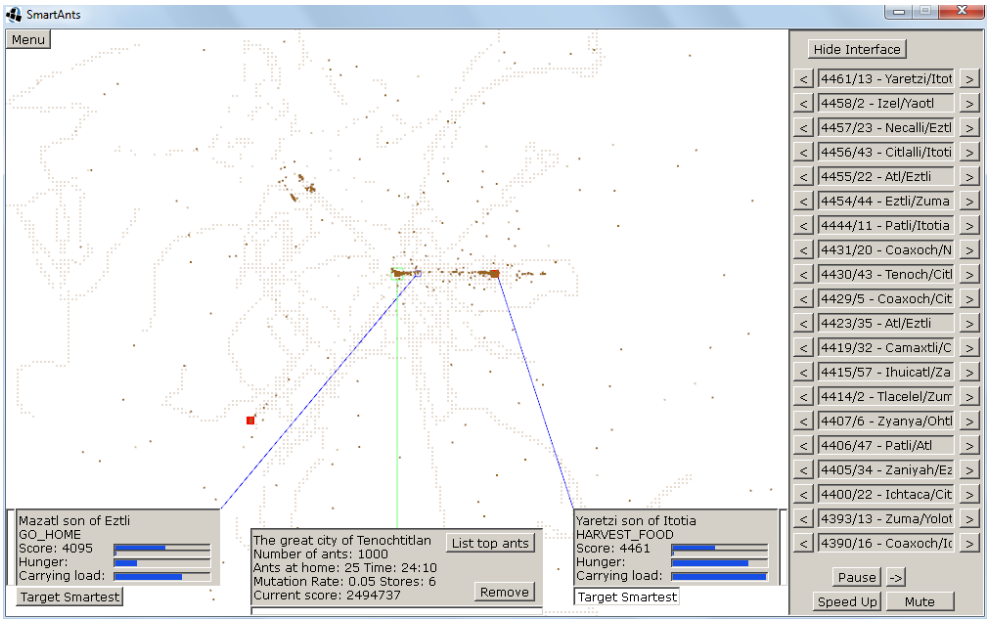
Arhitektura simulacijskega programa je sledeča: v glavnem razredu imamo tabelo kolonij, pri čemer ima vsaka kolonija svojo tabelo agentov. Vsak agent ima svoje možgane, predstavljene z nevronsko mrežo, ki je predstavljena s tabelo nevronov, ki so med seboj povezani z nevronskimi povezavami. Način zapisa nevrona in povezav med nevroni je opisan v poglavju 3.5.5. Za beleženje hrane in feromonov imamo še dodatna dva statična razreda s tabelami virov hrane oz. feromonov. Izris grafičnega okna s pripadajočimi gumbi, tekstom, vnosnimi polji in meniji nadzira poseben statični razred (Slika 3.1).



Slika 3.1: Arhitektura simulacijskega programa.

3.2 Grafično okno z uporabniškim vmesnikom

Da bi bilo možno realno časovno sledenje agentom, se v grafično okno izrisujejo elementi okolja v obliki različno pobarvanih kvadratov. Koloniji je določena naključna barva, v kateri se izriše dom, hrana pa je za boljšo prepoznavnost vedno rdeče barve. Agenti so izrisani v velikosti dveh točk v barvi svoje kolonije. Da bi ločili med agenti, ki nosijo hrano, in ostalimi, je privzeta barva agentov delno prosojna, tisti, ki nosijo hrano, pa imajo polno barvo. Tak način določanja barv omogoča prepoznavanje pripadnosti vsakega od agentov svoji koloniji. Agent lahko za seboj pušča t. i. feromon, ki je izrisan v velikosti ene točke v barvi kolonije, ki ji agent pripada, pri čemer je barva nekoliko prosojna. Dodan je tudi zvočni element, ki omogoča lažje spremljanje hitrosti odlaganja hrane doma. Tako simulacija zapiska vsakič, ko agent doma odloži hrano (Slika 3.2).



Slika 3.2: Grafično okno z uporabniškim vmesnikom.

Uporabniški vmesnik omogoča več možnosti. Na desni je izpisan seznam dvajsetih najbolj uspešnih agentov, ne glede na to, ali so že umrli.

Slika 3.3: Statistika kolonije.

V spodnjem delu uporabniškega vmesnika se nahaja okno s statistiko kolonije, ki zajema ime kolonije, število vseh agentov, število agentov doma, čas, stopnjo mutacije, trenutni skupni rezultat - seštevek rezultatov vseh živečih pripadnikov te kolonije (Slika 3.3).

V levem in desnem spodnjem delu vmesnika se nahajata dve okni, s katerima lahko sledimo poljubnemu agentu ali določimo, da želimo slediti najpametnejšemu agentu (*Target Smartest*). Za agenta, ki mu sledimo, se izpisujejo

Slika 3.4: Statistika agenta.

ime, trenutno vedenje, število točk, stopnja lakote in količina hrane, ki jo nosi (Slika 3.4).

Preko grafičnega vmesnika lahko vklopimo ali izklopimo zvok (*Mute*), pospešimo simulacijo (*Speed Up*), ki je implementirana tako, da se osveževanje grafičnega okna zmanjša in z miško približamo posamezen del simulacije.

Simulacija shranjuje podatke o agentih in kolonijah v datoteke, kar omogoča kasnejšo obdelavo podatkov in primerjavo vpliva različnih parametrov na dosežke in razvoj agentov.

3.3 Parametri simulacije in meni

Parametri simulacije so nastavljeni na vnaprej določene vrednosti, ki smo jih uporabili tudi pri analizi tipičnih vzorcev vedenja agentov in kolonij. Ti so podrobneje opisani v nadaljnjih poglavjih. V uporabniškem vmesniku lahko s pomočjo menija spreminjamo skoraj vse parametre simulacije (Slika 3.5). Te nastavitve lahko razdelimo v tri sklope:

a) Nastavitve, **vezane na kolonijo**, ki se upoštevajo ob dodajanju nove kolonije:

   - skupno število agentov;

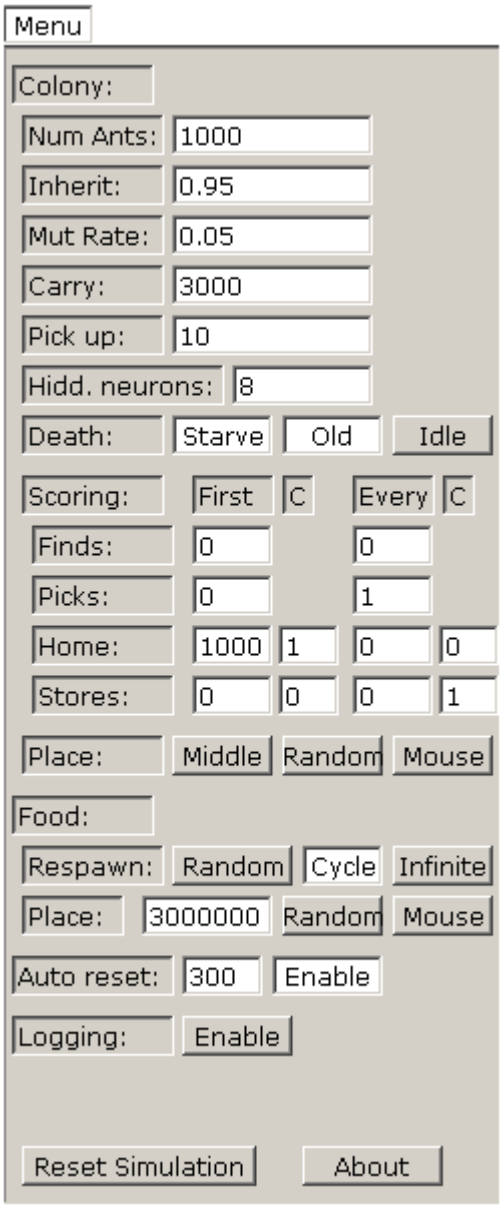
- verjetnost dedovanja (nasproti verjetnosti naključnega generiranja novega agenta);
- stopnja mutacije;
- maksimalna količina hrane, ki jo lahko agent nosi;
- maksimalna količina hrane, ki jo lahko agent nabere z enim nabiranjem;
- število nevronov v skritem nivoju nevronske mreže;
- razlogi za umiranje agentov (lakota, starost, neaktivnost);
- nastavitve točkovanja uspešnosti agentov glede na iskanje, pobiranje, prinašanje hrane domov in odlaganje hrane v zalogo s posameznimi nastavitvami za prvo in vsako nadaljnje vedenje ter obtežitev točk glede na količino hrane, ki jo agent nosi;
- pozicija nove kolonije (na sredini, naključno, na poljubni poziciji).

b) Nastavitve, **vezane na hrano**:

- končnost vira hrane (končen ali neskončen) in pojavljanje novega vira ob izčrpanju končnega vira (na naključni poziciji, na vnaprej določeni oddaljenosti od sredine simulacijskega okolja) za vse vire hrane v simulaciji;
- količina in pozicija novega vira hrane (naključno, na poljubni poziciji).

c) **Splošne** nastavitve:

- vklop/izklop ponovnega zagona simulacije na določeno število minut;
- vklop/izklop beleženja aktivnosti.



Slika 3.5: Meni za spreminjanje nastavitev simulacije.

3.4 Viri hrane

Za potrebe analize podatkov in izenačenost čim večjega števila spremenljivk med različnimi pogoji smo v simulaciji določili fiksno število virov hrane

in vnaprej določili njihove pozicije. Agenti imajo tako v vsakem trenutku na voljo dva vira hrane, ki se nahajata na dveh koncentričnih krožnicah s središčem v domu kolonije. Prva vira hrane sta zamaknjena za 135°. Ko hrane zmanjka, se na njegovi krožnici pojavi nov vir, ki je od prejšnjega oddaljen za 135° v smeri urinega kazalca. Spreminjanje pozicije hrane za več kot 90° omogoča, da se agenti ne morejo zanašati le na feromone drugih, ampak morajo vir hrane iskati vedno znova.

3.5 Agenti

3.5.1 Premikanje

Na agentovo premikanje vplivata dva vektorja, zapisana z decimalnimi števili, in sicer vektor pozicije in vektor premikanja. Prvi pove, kje se agent trenutno nahaja, drugi pa vsebuje informacijo o smeri in hitrosti premikanja. Hitrost premikanja je določena z dolžino vektorja premikanja in je omejena na vrednosti med 0 in 1. Ko agent izbere enega izmed štirih izhodov nevronske mreže, ki mu omogočajo premikanje (tj. hoja naokoli, hoja v smeri doma, hoja v smeri hrane ali sledenje feromonom), obstaja 10 % verjetnost, da bo spremenil svojo smer. Trenutnemu vektorju premikanja se prišteje naključni vektor z maksimalno dolžino 0,5. Vektor, ki ga prištevamo, je manjši od maksimalnega vektorja zato, da je premikanje in obračanje agenta bolj zvezno in s tem videti bolj naravno. Če je dolžina novega vektorja premikanja večja od 1, se ta normalizira. Če agent pri premikanju doseže rob simulacijskega okolja, se njegov vektor premikanja obrne proti domu in poveča na maksimalno hitrost.

3.5.2 Nabiranje hrane

Eden izmed izhodov agentovih možganov je nabiranje hrane (*GatherFood*). Ta omogoča, da ko agent stoji v neposredni bližini ali neposredno na lokaciji vira hrane, hrano tudi nabere. Agent ima vnaprej določeno kapaciteto hrane, ki jo lahko prenaša, in je nastavljena na 3000. Z vsakim nabiranjem hrane agent naloži le manjšo količino hrane (10), zato je količina hrane, ki jo agenti nosijo, zelo različna. Da bi agent nabral polno količino hrane, ki jo je zmožen nositi, se mora za nabiranje hrane odločiti večkrat zapored (300-krat). Količina hrane, ki jo agent nosi, predstavlja tudi enega izmed vhodov v agentovo nevronsko mrežo.

3.5.3 Količina hrane in nadziranje prenosa med viri hrane in agenti

Hrana se lahko prenaša od vira hrane ali od zaloge do agenta ali pa od agenta v zalogo. Podatkovno strukturo, ki skrbi za nadzor nad količino hrane in za nadzor prenosa hrane, tvori razred *FoodSource*. Ta ima spremenljivko *size*, v kateri so zapisane trenutna količina hrane in dve funkciji za prenašanje in odvzemanje hrane.

Funkcija za prenašanje (*transferFromOtherSource()*) sprejema dva parametra: vir (*FoodSource*), iz katerega se bo hrana prenašala, in želeno količino prenesene hrane. Da ne bi prišlo do prenosa hrane, ki ni na voljo, funkcija najprej preveri, če je v viru zadostna količina hrane za prenos. Viru nato trenutno količino hrane (*size*) zmanjša za velikost prenosa, svoji spremenljivki *size* pa trenutno količino hrane poveča za isto vrednost.

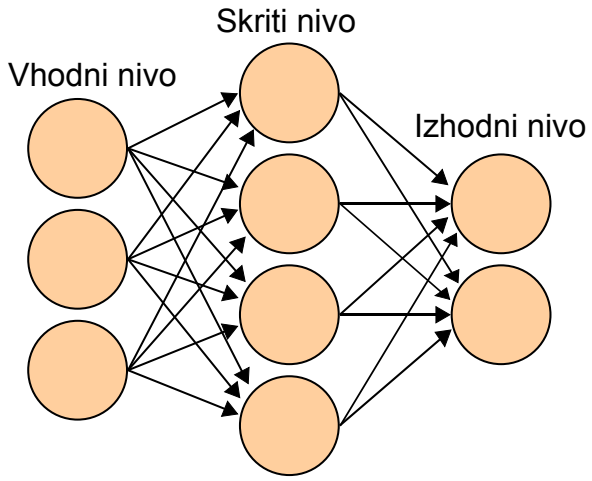
V primeru hranjenja ne prihaja do prenosa hrane, ampak hrano le odvzemamo. Uporabljamo podobno funkcijo kot v zgornjem primeru, pri čemer se hrana le izniči. Funkcija za odvzemanje hrane (*take()*) sprejema le en pa-

rameter, ki predstavlja količino hrane, ki si jo želimo viru odvzeti, in vrača vrednost hrane, ki jo je agent uspešno pojedel. Ob tem zmanjša vrednost spremenljivki *size*.

3.5.4 Hranjenje

Agent ima svojo zalogo energije, ki jo pridobiva s hranjenjem. Ta je nastavljena na 80, pri čemer z vsako akcijo hranjenja porabi dve enoti hrane in si s tem pridobi dve enoti energije. Agent se lahko hrani s hrano, ki jo nosi, ali s hrano, ki je shranjena doma. Za slednjo se mora trenutno nahajati na področju doma. Če agentova energija pade na 0, agent zaradi lakote umre.

3.5.5 Možgani



Slika 3.6: Primer nevronske mreže [12].

Možgani vsakega agenta so implementirani z enosmerno, trinivojsko, polno povezano nevronsko mrežo. Prvi nivo nevronov predstavlja šest vhodov, skriti nivo ima osem nevronov, zadnji, tretji nivo, pa predstavlja osem izho-

dnih nevronov, kar sledi priporočilom, opisanim v poglavju 1.2. Nevroni na sosednjih nivojih so med seboj povezani (Slika 3.6).

Nevron je predstavljen s svojim razredom. Glavne podatkovne strukture, ki jih vsebuje, so njegova vrednost, ki je lahko med 0 in 1, in dva seznama nevronskih povezav. Prvi seznam vsebuje povezave, ki kažejo na nevron, drugi seznam pa vsebuje povezave, ki vodijo iz nevrona. Ker se vrednost nevrona konstantno spreminja, vsebuje tudi funkcijo za izračun svoje vrednosti.

Pri nevronih na prvem (vhodnem) nivoju predstavljajo vrednosti vhodne podatke, ki jih agent o sebi in svoji okolici dobi vsak trenutek. Nekateri vhodni podatki so binarni (smiselni sta le vrednosti 0 in 1; npr., ali se agent nahaja doma), drugi pa so številski in lahko zavzemajo vse vrednosti med 0 in 1 (npr. stopnja lakote). Nevroni na drugem, skritem nivoju, in na tretjem, izhodnem nivoju, izračunajo svojo vrednost tako, da iterirajo po seznamu svojih vhodnih povezav, pri čemer za vsako povezavo zmnožijo njeno vrednost z vrednostjo nevrona na drugi strani povezave, rezultate vseh zmnožkov pa seštejejo.

Tudi nevronska povezava ima lastni razred, ki vsebuje izvorni in ponorni nevron. Utež je vrednost med 0 in 1, pri čemer je pomembno, da se v nasprotju z vrednostmi nevronov vrednost uteži tekom simulacije ne spreminja. Določi se ob kreaciji možganov in s tem omogoča in vzdržuje individualnost vsakega agenta. Uteži je potrebno normalizirati tako, da je seštevek uteži vseh povezav, ki kažejo na nek nevron, enak 1.

3.5.6 Umiranje

Kolonija ima vedno enako število agentov. Ti umirajo zaradi starosti ali zaradi lakote, kar omogoča, da se rojevajo novi in se tako razvoj kolonije nadaljuje. Lakoto in starost predstavljata dve spremenljivki, ki imata določeni začetni vrednosti. Med vsako iteracijo obstaja majhna verjetnost, da

se bosta vrednosti teh dveh spremenljivk zmanjšali. Ko ena od njiju doseže vrednost 0, agent umre. Lakota predstavlja kratkoročno stanje, ki ga lahko agenti zaznajo in na katerega lahko vplivajo, na starost pa agenti nimajo vpliva in ima kot dolgoročno posledico smrt. Spremenljivka, ki predstavlja starost, je zato ob začetku simulacije veliko večja kot vrednost spremenljivke, ki predstavlja lakoto.

3.5.7 Dedovanje in mutacija

Da bi omogočili variabilnost agentov in s tem možnost napredka kolonije, uporabljamo dva mehanizma, in sicer mutacijo že obstoječih nevronskih mrež ter možnost naključnega generiranja agenta. Ob kreaciji novega agenta ta deduje možgane enega od dvajsetih najbolj uspešnih agentov, četudi je ta že umrl. Več točk, kot ima agent na lestvici najbolj uspešnih, večja je verjetnost, da bodo novi agenti podedovali njegove možgane, kar smo implementirali s pomočjo uteženega naključja – novemu agentu se določi naključna vrednost med 0 in skupnim seštevkom vseh točk z lestvice najboljših, nato pa v vrstnem redu od najboljšega do najslabšega agenta izbere tistega, katerega kumulativna vrednost točk je večja od izbrane vrednosti novega agenta. Pri dedovanju je upoštevana določena stopnja mutacije, ki smo jo za potrebe analize podatkov za vsako posamezno kolonijo določili med 0,01 in 0,10 v korakih po 0,01. Vsaki uteži podedovane nevronske povezave se tako pri dedovanju odšteje ali prišteje vrednost med nič in stopnjo mutacije. Na koncu uteži vseh vhodnih povezav do posameznega nevrona normaliziramo, tako da je njihov seštevek ponovno 1. Poleg opisanih mutacij je ob kreaciji agenta 5 % možnost, da ta nevronske mreže ne bo dedoval, ampak bodo povezave v njegovi nevronski mreži vsebovale popolnoma naključne vrednosti. S tem zagotovimo, da se v primeru velikega števila slabo razvitih agentov generirajo tudi novi, drugačni agenti, tako da kolonija ne stagnira.

3.6 Feromoni

Ko agenti poberejo in nosijo hrano, za seboj puščajo feromonske sledi, tako da lahko ostali agenti tem feromonom sledijo in tako pridejo do vira hrane. Tekom izdelave simulacije smo preizkusili različne načine implementacije zapisa in uporabe feromonov in med njimi poiskali rešitev, ki je delovala dovolj hitro in je bila tudi vsebinsko primerna. Agent kot vhodni podatek pridobi informacijo, ali trenutno v svoji okolici zaznava feromone ali ne. Če ima nevronska mreža izhod *FOLLOW_PHEROMONES*, začne agent slediti feromonski sledi.

3.6.1 Zapis feromona

Feromoni, ki jih agenti puščajo, imajo različno moč, kar omogoča drugim, da lahko med več feromoni v njihovi okolici izberejo močnejšega. Ko agent pobere hrano, začne puščati feromonske sledi. Moč feromonov, ki jih pušča, se s časom zmanjšuje, prav tako pa se zmanjšuje moč že izpuščenega feromona, kar predstavlja njegovo izhlapevanje. Ko agent doma odda hrano, pade moč izpuščanja feromonov ponovno na nič. Če agent izpusti feromon na koordinatah, kjer se že nahaja feromon, ga prepiše le v primeru, če je njegova moč večja od že obstoječega feromona.

V prvi iteraciji načrtovanja okolja smo zmanjševali moč izpuščenega in že postavljenega feromona za ena. Posledično je imela skoraj celotna feromonska sled enako moč feromonov, saj so se vsi enakomerno in naenkrat zmanjševali. V drugi iteraciji so feromoni izhlapevali za ena, moč izpuščenih feromonov pa se je zmanjševala za dva. Tudi ta način ni bil ustrezen, saj se agenti velikokrat tudi ustavijo in ne puščajo sledi z enakomerno padajočo močjo, hkrati pa lahko drugi agenti pohodijo in prepišejo njihovo sled. Tako se lahko v okolici agenta na različnih mestih nahaja več enako močnih feromonov, zato

so se agenti, ki so takšni sledi sledili, velikokrat premikali le naprej in nazaj oz. popolnoma obstali na mestu.

V tretji iteraciji smo skušali zgoraj opisani problem rešiti s pomočjo uvedbe vektorjev. Vsak feromon je imel dve vrednosti – moč in vektor, ki je kazal v nasprotni smeri od smeri agenta, ki ga je izpustil. Če je bilo več enako močnih feromonov v agentovi okolici, se je ta še vedno premikal v smeri hrane, saj so tja kazali vsi vektorji izpuščenih feromonov. Ta rešitev je sicer ustrezna, vendar pa ne upošteva več primarne funkcije feromona, kot ga poznamo v naravi, kjer feromon v osnovi le označuje določeno mesto in ne podaja smeri, iz katere je prihajal agent, ki ga je izpustil.

V četrti iteraciji smo zato poiskali drugačno rešitev, ki vektorjev ni vsebovala. Moč feromona smo zmanjševali le pri njegovem izpustu, izhlapevanje feromona pa smo namesto s padanjem moči simulirali le s pomočjo odštevalnika časa tako, da je po določenem času feromon izginil. Prepis feromonov deluje podobno kot v prvi iteraciji, le da smo v primeru podobne moči obeh feromonov že obstoječemu feromonu povečali njegov čas obstoja.

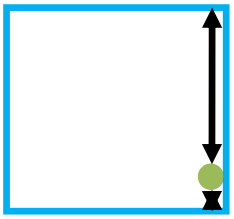
3.6.2 Prepoznavanje feromonov

V prvi iteraciji implementacije prepoznavanja feromonov smo feromone zapisali s pomočjo razpršene tabele (*hashtable*), v kateri je vrednost ključa predstavljal vektor s koordinatama x in y. Vsak agent v vsaki iteraciji za vsako točko v radiju 10 točk okoli sebe preveri, ali se tam nahaja feromon. Rešitev je sicer vsebinsko primerna, vendar je delovala počasi.

V drugi iteraciji smo dodali še eno razpršeno tabelo, ki vsebuje več manjših razpršenih tabel, ki predstavljajo okolje, razdeljeno na mrežo kvadratov velikosti 20 x 20 točk. Ključ za dostop do razpršenih tabel se računa po enačbi 3.1, ključ znotraj manjših tabel pa je vektor x, y, tako kot v prvi iteraciji.

$$\frac{x}{20} * 1000 + \frac{y}{20} \tag{3.1}$$

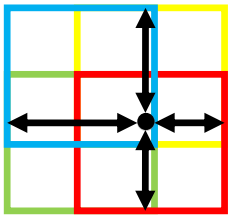
V postopku preverjanja, ali so feromoni v okolici agenta, agent preveri, ali kvadrat, v katerem se nahaja, vsebuje feromone (oz. ali je razpršena tabela prazna ali polna). Če je polna, agent znotraj te tabele med feromoni, ki se tu nahajajo, poišče feromon z največjo močjo. Tovrstni pristop je časovno bolj ekonomičen, vendar pa prihaja do težav, ko se agent nahaja ob robu kvadrata. V najslabšem primeru se lahko namreč zgodi, da v neko smer ne zaznava niti ene točke oddaljenosti (Slika 3.7).



Slika 3.7: Prikaza agenta znotraj kvadrata v mreži in problem asimetričnega zaznavanja feromonov.

V tretji iteraciji smo okolje razdelili na štiri mreže, ki se med seboj delno prekrivajo in ustvarjajo kvadrate velikosti 10 x 10 točk (Slika 3.8). Agent tako zaznava feromone v vsako smer najmanj 10 točk in največ 20 točk. V postopku iskanja feromonov agent za vsako od štirih razpršenih tabel, v kateri se nahaja, preveri, ali je prazna oz. polna in če sledi feromonski sledi, poišče najmočnejši feromon. Ta način se je izkazal za zelo učinkovitega tako s časovnega kot vsebinskega vidika.

Če se je v posameznem kvadratu nahajalo veliko število agentov in so le-ti puščali feromonske sledi, je lahko vsak kvadrat vseboval do 400 feromonov. Tak način izračuna in izrisovanja feromonov je v realno časovni simulaciji deloval dobro. Če smo želeli simulacijo pospešiti, da bi lahko v krajšem času pridobili več podatkov za analizo, smo morali ta del simulacije še dodatno



Slika 3.8: Elementi štirih mrež, ki se med seboj prekrivajo, ter agent na sredini. Puščice prikazujejo, kako daleč agent zaznava feromone v vsaki izmed smeri.

pospešiti. Tako smo feromone shranjevali in izrisovali le na koordinatah, ki so večkratniki števila štiri, kar je število možnih zapisov in izrisov znotraj enega kvadrata zmanjšalo na 25. To je celotno simulacijo zelo pospešilo, hkrati pa ni opazno vplivalo na delovanje feromonov.

3.7 Točkovanje

Agenti v večini primerov dedujejo svoje možgane od enega izmed dvajsetih najboljših pripadnikov kolonije, pri čemer je verjetnost dedovanja agentovih možganov proporcionalna njegovemu številu točk v primerjavi z drugimi agenti na lestvici. Agente nagrajujemo le za vedenja, ki omogočajo koloniji preživetje, za nepravilno vedenje pa agentov ne kaznujemo. Na začetku ima vsak agent 0 točk, pridobljene točke pa se nabirajo kumulativno vse do njegove smrti. Agent pridobiva točke na tri različne načine, ki so glede na pomembnost akcije različno obteženi (ker večje število točk pri agentu pomeni tudi večjo verjetnost dedovanja njegovih možganov).

Bistveno za agentovo preživetje je nabiranje hrane, zato je agent nagrajen za aktivnosti, ki so vezane na nabiranje hrane.

a) Ko agent nabira hrano, dobi točke glede na količino pobrane hrane.

Naenkrat lahko dobi maksimalno 300 točk (kar je 1 točka na 10 enot hrane pri maksimalni kapaciteti 3000 enot hrane). S tem spodbudimo agente, da pobirajo hrano, ko pridejo do nje.

Ena izmed ključnih značilnosti preživetja kolonije je skladiščenje hrane doma. Zato so agenti nagrajeni za tako vedenje.

b) Ko prvič v svojem življenju pride domov s hrano, dobi agent 1000 točk in dodatno točko za vsako enoto hrane, ki jo v tistem trenutku nosi (torej med 1-3000 točk). V nasprotju z drugima dvema načinoma točkovanja se to točkovanje izvede le enkrat v agentovem življenjskem obdobju. Namenjeno je spodbujanju prehoda med individualnim vedenjem agentov in usmerjenostji k preživetju kolonije kot celote.

c) Vsakič, ko agent nabrano hrano odloži doma, dobi točko za vsako odloženo enoto hrane (med 1 in 3000 točkami).

Za agentovo preživetje je pomembno tudi hranjenje, ki pa ni neposredno nagrajeno s točkami, saj je nagrajen posredno, ker lahko v primeru, da se ne hrani učinkovito, umre zaradi lakote in tako zbere manjše število točk, kot bi jih tekom svojega življenja lahko nabral sicer.

3.8 Časovnik

Pogostost izvedbe posameznih delov programa v začetni fazi razvoja simulacije ni bila omejena. Ko se je, zaradi povečanja kompleksnosti simulacije, pojavila potreba po pohitritvah, smo uvedli časovno omejevanje določenih funkcij. Bolj računsko zahtevne funkcije smo omejili tako, da so se smele izvesti samo vsako določeno število milisekund.

V zadnjih iteracijah simulacije smo implementirali tudi pridobivanje podatkov, kar je zahtevalo pospešitev simulacije, saj bi bilo v nasprotnem primeru realno časovno pridobivanje podatkov preveč zahtevno. Tako je bilo potrebno stari sistem omejevanja v milisekundah spremeniti. Ker smo si želeli čim bolj pospešiti čas znotraj simulacije, pospešitev pa je nihala s trenutno računsko zahtevnostjo, za nadzor funkcij simulacije ni bilo več mogoče uporabljati realnih časovnih enot.

Uvedli smo virtualno časovno enoto okvirja (*frame*), ki je bila enaka hitrosti osveževanja naše simulacije. V eni sekundi se glavna zanka simulacije izvede 60-krat, vsakič ko se ta zanka izvede, pa trenutno število okvirjev povečamo za ena. To število je shranjeno v posebnem razredu *TimerControl*, ki vsebuje spremenljivke, ki povedo, na koliko časovnih enot se določene funkcije izvajajo. Te funkcije ob vsaki iteraciji glavne zanke preverijo, ali se smejo izvesti. Pogostost izvajanja posamezne funkcije je sledeča:

- izrisovanje grafičnih elementov – vsako časovno enoto;
- agenti (funkcije, kot so npr. premikanje, nabiranje) – vsaki dve časovni enoti;
- osveževanje in odstranjevanje feromonskih sledi – vsaki dve časovni enoti;
- osveževanje nevronske mreže – vsakih 12 časovnih enot;
- osveževanje in odstranjevanje posameznih virov hrane – vsakih 20 časovnih enot.

Na ta način lahko simulacijo pospešimo (ali pa upočasnimo) kolikor želimo, pri čemer lahko še vedno sledimo količini simuliranega realnega časa, saj sproti štejemo število iteracij glavne zanke simulacije.

Za ozko grlo pri hitrosti izvajanja simulacije se je izkazalo izrisovanje grafičnih elementov, ki so se izrisovali ob vsaki iteraciji glavne zanke. To je bilo pri pridobivanju podatkov sila neučinkovito.

Z novim implementiranim sistemom okvirjev smo nastavili grafično izrisovanje na vsakih 300 izvedb glavne zanke oz. časovnih enot, kar je vsakih 5 sekund časa znotraj simulacije. S to spremembo simulacija teče tudi do 20-krat hitreje kot prej, pri čemer je pospešitev vidna predvsem na začetku, ko agenti še ne nabirajo hrane in ne puščajo feromonskih sledi.

Da lahko agente in njihovo nabiranje hrane še vedno opazujemo v realnem času, je v uporabniškem vmesniku gumb, s katerim preklapljamo med normalnim in hitrim načinom simulacije.

3.9 Beleženje aktivnosti

S statičnim razredom *Logger* smo implementirali beleženje večjega števila spremenljivk, kar omogoča kasnejšo obdelavo podatkov in ugotavljanje vpliva parametrov na učinkovitost agentov in s tem kolonije. Vse spremenljivke beležimo enkrat na minuto, pri čemer poteka pridobivanje podatkov za eno kolonijo 300 minut (5 ur). Kot neodvisno spremenljivko smo izbrali delež mutacije (vrednosti mutacije so od 0,01 do 0,10, označene pa so s kategorijami od 1 do 10; *MutationRate*). Čeprav pri obstoječi verziji simulacije to ni potrebno, beležimo tudi skupno število živečih agentov (*Total*), kar bi omogočalo primerjavo različno velikih kolonij.

V posebnem statičnem razredu beležimo naslednjih deset odvisnih spremenljivk:

a) kolikokrat so agenti odložili hrano doma v zadnji minuti (*Stores*),

b) kolikokrat so agenti odložili hrano doma od začetka obstoja kolonije

(*TotalStores*),

c) število točk najboljšega agenta (ne glede na to, ali je živ ali ne; *BestAntScore*),

d) število točk najboljšega živega agenta (*BestAliveAntScore*),

e) skupni seštevek točk dvajsetih najboljših agentov (*TopAntsScoreSum*),

f) skupni seštevek točk za vse živeče agente (*AllAliveAntsScoreSum*),

g) število produktivnih agentov v zadnji minuti (agenti, ki so v zadnji minuti domov prinesli hrano; *Productive*),

h) število agentov, ki v zadnji minuti niso za vsaj pet sekund zapustili svojega doma (*Useless*),

i) koliko bližnjih oz. daljnih virov hrane je bilo izčrpanih do sedaj (*InnerGathered*, *OuterGathered*) in

j) povprečna starost trenutno živečih agentov (*AverageLife*).

Vse ostale spremenljivke, ki bi lahko vplivale na učinkovitost kolonije, smo nadzorovali tako, da so bile izenačene (npr. količina in pozicija virov hrane, število agentov v koloniji, pozicija kolonije itd.).

Simulacija zapisuje in razporeja podatke v več datotek glede na stopnjo mutacije. Podatki za kolonije, ki so obstajale manj kot celotni čas simulacije, se ne zapišejo. Pridobivanje podatkov za večjo količino kolonij je potekalo avtomatično, saj se je simulacija po preteku 300 minut samodejno prekinila, ponastavila vse spremenljivke in se ponovno zagnala.

4. Tipični vzorci vedenja

Po končani izdelavi simulacije nas je zanimalo, kako uspešne so posamezne kolonije glede na stopnjo mutacije pri dedovanju in kakšni so njihovi tipični vzorci vedenja. Stopnjo mutacije smo označili z vrednostmi od 0,01 do 0,10 (npr. pri stopnji mutacije 0,01 se uteži na nevronskih povezavah pri dedovanju spreminjajo največ za $\pm 0,01$). Za vsako stopnjo mutacije smo zbrali podatke za 24 med seboj neodvisnih kolonij (skupno torej 240 kolonij) v dolžini 300 minut. Vrednosti, ki smo jih beležili, so opisane v poglavju 3.9.

Opazovanje tipičnega obnašanja lahko razdelimo v grobem na dva dela. V prvem delu nas je zanimalo, kakšno je obnašanje agentov, v drugem delu, kakšno je obnašanje agentov z vidika preživetja kolonije, na koncu pa smo naredili še primerjavo najboljših agentov glede na stopnjo mutacije.

4.1 Vedenje agentov

Za obstoj agentov znotraj kolonije je pomembno, da najdejo hrano, jo poberejo in pojejo. Prvi dve vedenji sta razvidni iz dveh spremenljivk, ki beležita, koliko bližnjih oz. daljnih virov hrane je bilo izčrpanih do sedaj (*InnerGathered*, *OuterGathered*), zadnja pa je posredno razvidna iz povprečne starosti trenutno živečih agentov (*AverageLife*).

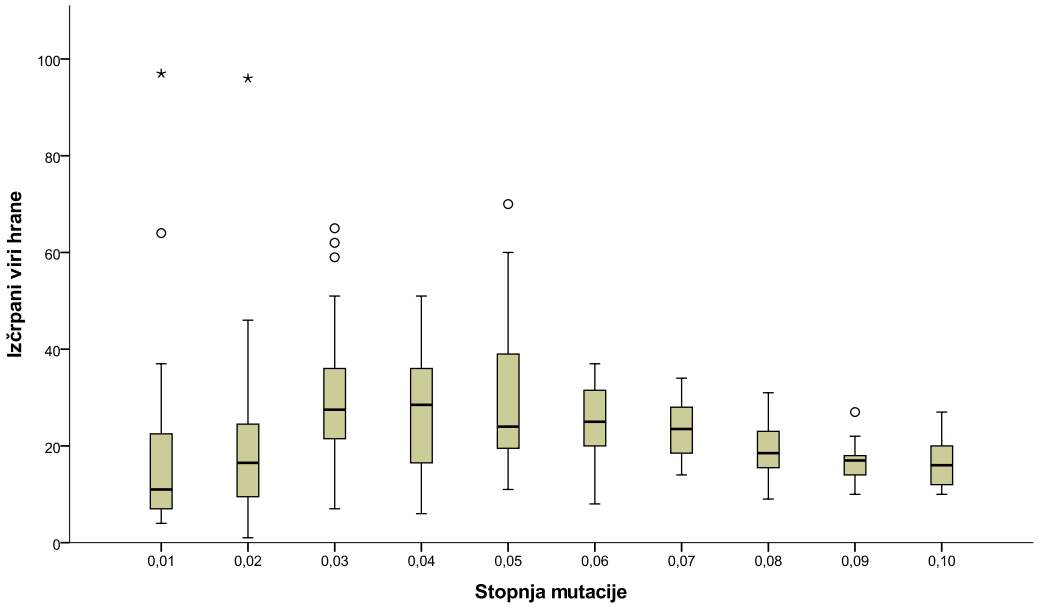
4.1.1 Število izčrpanih virov hrane

Za prikaz odnosa med stopnjo mutacije in številom izčrpanih virov hrane v celotnem času obstoja kolonije (ne glede na to, ali je bila hrana bližnja ali daljna - *TotalGathered*) smo uporabili škatlo z brki (*boxplot*; Slika 4.1). Zgornji in spodnji del škatle prikazujeta prvi in tretji kvartil - vrednosti, pod katerima je 25% oz. 75% primerov. Srednja odebeljena črta predstavlja mediano, rob črt izven škatle pa minimalno in maksimalno vrednost. Vrednosti, ki zelo odstopajo od preostalih podatkov (t. i. *outliers*), so na grafu označene s krogci in zvezdicami.

Rezultati kažejo, da so imele kolonije s stopnjo mutacije 0,01 in 0,02 v primerjavi z drugimi kolonijami v povprečju nižje število izčrpanih virov hrane, kar je najverjetneje posledica tega, da je stopnja mutacije v teh dveh primerih še premajhna, da bi se lahko kolonija neučinkovitih agentov zanesljivo razvila v uspešno delujočo kolonijo. Vendar pa takšna majhna stopnja mutacije hkrati omogoča koloniji, da v redkih primerih, ko se v njej pojavi zelo uspešen agent, tega zvesto kopira. Tako lahko na grafu vidimo, da dosegajo kolonije najvišje zabeleženo število izčrpanih virov hrane prav pri najnižjih stopnjah mutacije (označene z zvezdico), pri višjih stopnjah mutacije pa do tako visokih odstopanj ne prihaja.

Pri stopnji mutacije, višji ali enaki 0,06 so dosežene vrednosti nizke in se ne spreminjajo dosti. Predvidevamo lahko, da so stopnje mutacije v teh primerih že prevelike, da bi se nevronske mreže uspešnih agentov brez velikih odstopanj prenesle na njihove potomce.

Glede na dobljene rezultate sklepamo, da je za doseganje učinkovitega pobiranja hrane znotraj kolonije najboljša stopnja mutacije med 0,03 in 0,05. Te stopnje mutacije omogočajo najbolj optimalno ravnovesje med zmožnostjo razvoja novih agentov, ko kolonija še ni uspešna, in zmožnostjo (skorajšnjega) kopiranja že uspešnih agentov.

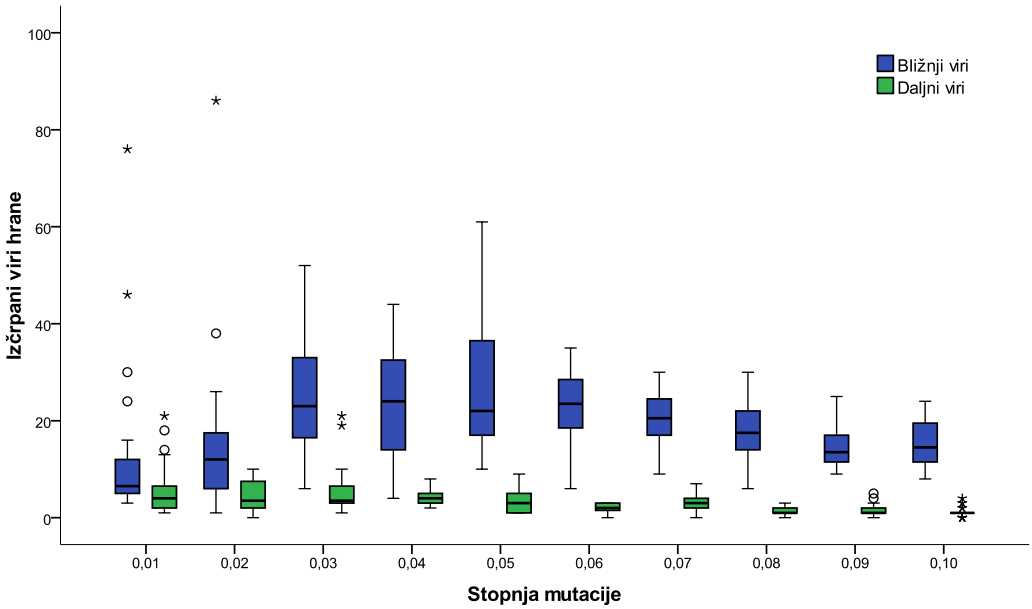


Slika 4.1: Porazdelitev skupnega števila izčrpanih virov hrane glede na stopnjo mutacije.

Na sliki 4.2 so posamezni viri hrane razdeljeni na bližnje (modre škatle) in daljne (zelene škatle). Kot bi lahko pričakovali, so agenti v veliko večji meri nabirali hrano bližje domu, ne glede na to, kakšna je bila stopnja mutacije kolonije. Takšno obnašanje je posledica feromonov in prednastavljene preference agentov za sledenje močnejšim feromonskim sledem. Ko agent pobere hrano, začne puščati feromonsko sled z določeno močjo, ki s časom upada. Tako je feromonska sled, ki vodi od bližnje hrane, vedno močnejša kot sled, ki vodi od bolj oddaljene hrane.

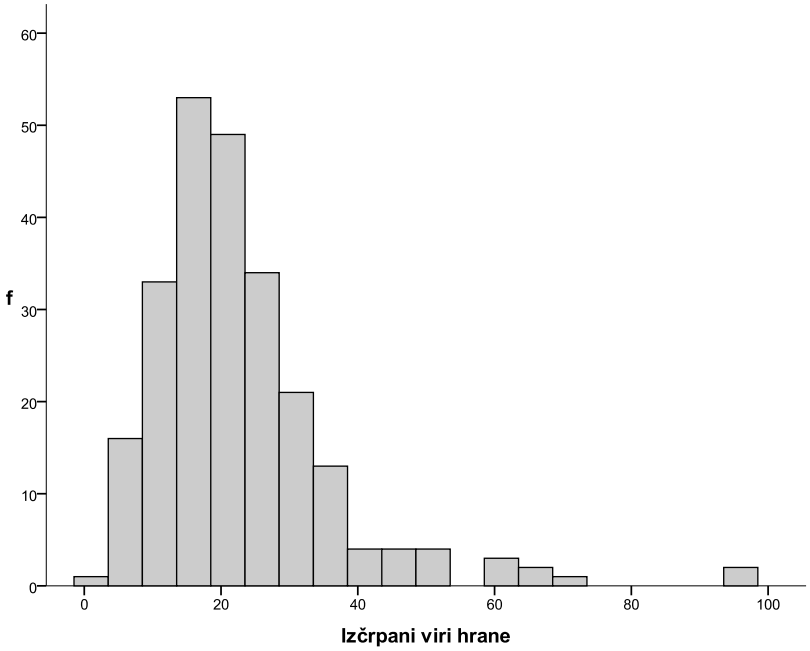
Večina kolonij je v danem časovnem okvirju izčrpala med 0 in 40 virov hrane (M = 22,71), le redke pa so izčrpale več kot 40 virov. Dve koloniji izrazito izstopata, saj sta uspeli izčrpati 96 oz. 97 virov hrane (Slika 4.3). Kot je razvidno iz slike 4.1, sta to koloniji s stopnjo mutacije 0,01 in 0,02.

Kolonije so za izčrpanje prvega vira hrane v povprečju porabile 25,68 minut



Slika 4.2: Porazdelitev števila izčrpanih bližnjih (modra barva) in daljnih (zelena barva) virov hrane glede na stopnjo mutacije kolonij.

(SD = 28,29). Kot je razvidno iz Tabele 4.1, se povprečni čas razlikuje tudi glede na stopnjo mutacije kolonij – največ časa so porabile kolonije s stopnjo mutacije 0,01 (51 minut), najmanj pa kolonije s stopnjo mutacije 0,05 in 0,10 (18 minut). Največjo razpršenost imajo kolonije s stopnjo mutacije 0,01, najmanjšo pa kolonije s stopnjo mutacije 0,05, 0,09 in 0,10, kar kaže na to, da so kolonije z nizko stopnjo mutacije precej bolj raznolike v uspešnosti pobiranja hrane kot kolonije z višjimi stopnjami mutacije.



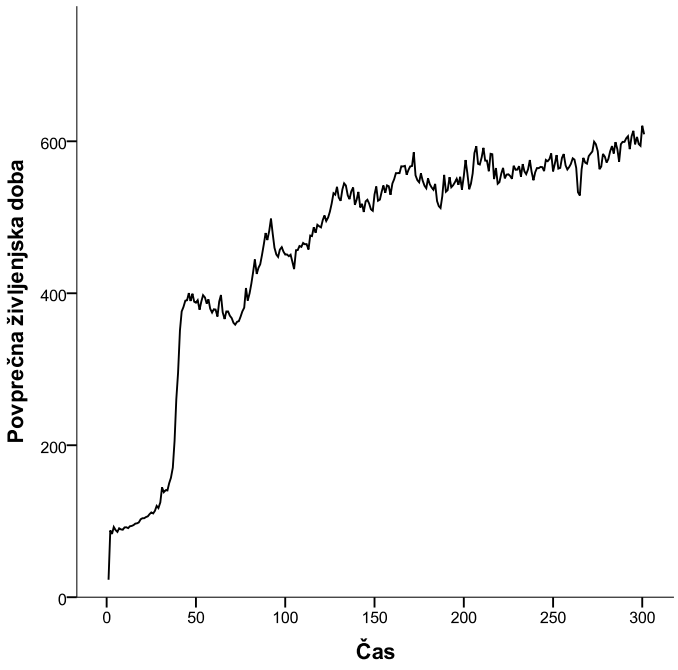
Slika 4.3: Porazdelitev skupnega števila izčrpanih virov hrane za vse kolonije (N = 240).

Stopnja mutacije	N	Minimum	Maksimum	M	SD
0,01	24	5	215	50,54	57,40
0,02	24	8	207	38,83	43,83
0,03	24	5	171	31,17	33,44
0,04	24	7	61	20,79	15,63
0,05	24	7	33	17,83	8,46
0,06	24	8	65	20,38	14,10
0,07	24	10	50	20,46	11,34
0,08	24	10	62	19,58	11,30
0,09	24	10	39	19,21	7,17
0,10	24	12	24	18,00	3,71

Tabela 4.1: Povprečni čas, porabljen za izčrpanje prvega vira hrane glede na stopnjo mutacije.

4.1.2 Povprečna starost

Čeprav nismo beležili količine hrane, ki jo agent poje, je to posredno razvidno iz agentove starosti. Agenti, ki hranjenja ne osvojijo, umrejo precej hitreje kot agenti, ki se uspešno hranijo. Prvih nekaj minut starost agentov znotraj kolonije raste, saj ti še ne umirajo zaradi lakote. Prvi večji skok v povprečni starosti je razviden šele okoli 50. minute, kar je najverjetneje posledica uspešnega hranjenja agentov (Slika 4.4).



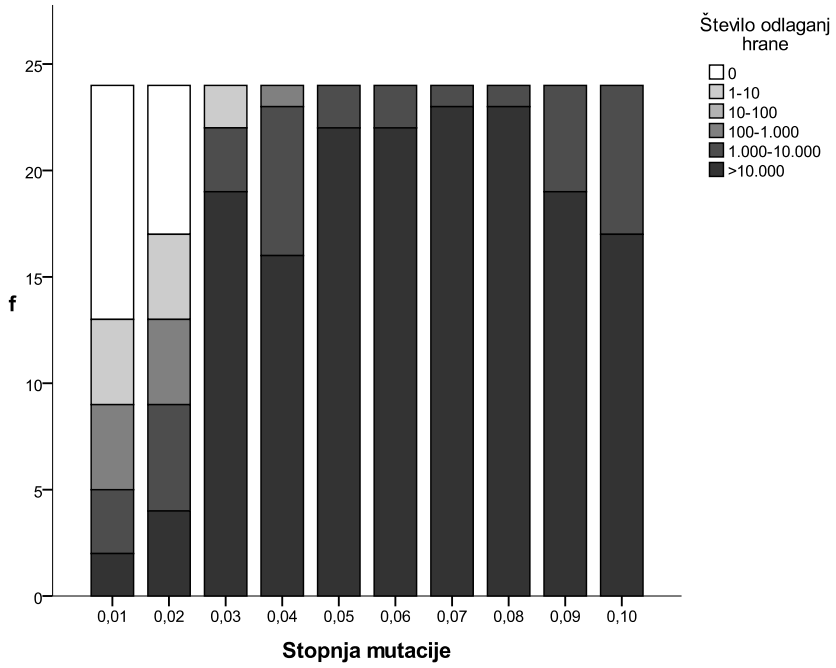
Slika 4.4: Povprečna starost agentov glede na čas za vse kolonije (N = 240).

4.2 Vedenje kolonije

Iz vidika preživetja kolonije kot celote je pomembno, ali agenti doma shranjujejo hrano ali ne. S tem povečajo možnost za hranjenje agentov, saj sta

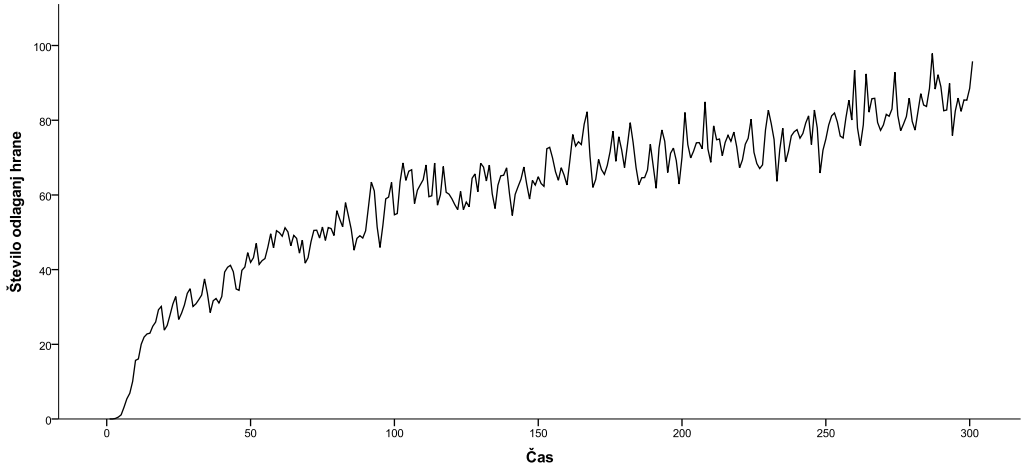
prehranjevanju namenjena dva izhoda iz nevronske mreže - hranjenje s hrano, ki jo agent nosi (EAT_OWN_FOOD), in hranjenje s hrano, ki je shranjena doma in je v skupni uporabi kolonije (EAT_COLONY_FOOD).

Nekatere kolonije se v celotnem času obstoja niso naučile odlagati hrane doma. Če razdelimo kolonije glede na stopnjo mutacije, lahko vidimo, da so tovrstne kolonije prisotne le pri stopnjah mutacije 0,01 in 0,02. Kolonije s stopnjo mutacije 0,04 imajo število odlaganj hrane vedno večje od 100, kolonije s stopnjo mutacije 0,05 in več pa imajo število odlaganj hrane vedno večje od 1000. Tako lahko vidimo, da kolonije z dovolj veliko stopnjo mutacije vedno razvijejo sistem shranjevanja hrane doma, pri čemer so najbolj uspešne kolonije s stopnjo mutacije med 0,05 in 0,08 (Slika 4.5).



Slika 4.5: Skupno število odlaganj hrane v celotnem času obstoja kolonije glede na stopnjo mutacije (N = 240).

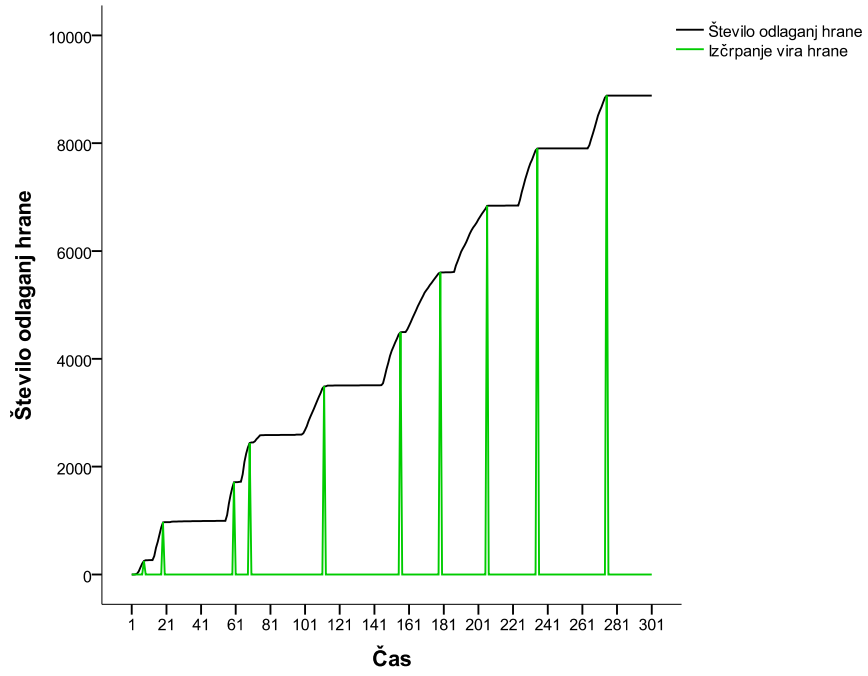
Kot bi lahko pričakovali, povprečno število odlaganj hrane s časom raste.



Slika 4.6: Povprečno število odlaganj hrane glede na čas za vse kolonije (N = 240).

Opazno je nihanje, ki je posledica izčrpanja trenutnega vira hrane (Slika 4.6). V tem času agenti prinesejo domov manj hrane, saj morajo med tem poiskati nov vir. Da bi lahko videli, kakšen je odnos med odlaganjem hrane in izčrpanjem virov, smo si ogledali grafe za posamezne kolonije. Enega izmed tipičnih grafov smo prikazali na sliki 4.7. Videti je, da količina odložene hrane raste, vse dokler se trenutni vir hrane ne izčrpa. Nato količina odložene hrane stagnira, vse dokler agenti ne odkrijejo novega vira hrane, kar je videti v skoku števila odlaganj hrane, dokler vir hrane ni ponovno izčrpan.

Razlike v skupnem številu odlaganj hrane med različnimi kolonijami so med drugim posledica razlik v številu izčrpanih virov hrane (oz. uspešnega iskanja in nabiranja hrane). Tudi izračun korelacije je pokazal, da je število odlaganj hrane pozitivno povezano s številom izčrpanih virov hrane (r = 0,768). Vendar pa korelacija ni popolna, kar kaže na to, da agenti v določenih primerih hrane ne prinesejo domov. Iz Tabele 4.2 vidimo, da na ta odnos vpliva tudi stopnja mutacije, saj je delež pojasnjene variance pri stopnji mutacije 0,01 zelo majhen (0,18), pri srednjih stopnjah mutacije pa je precej večji (okoli 0,80).

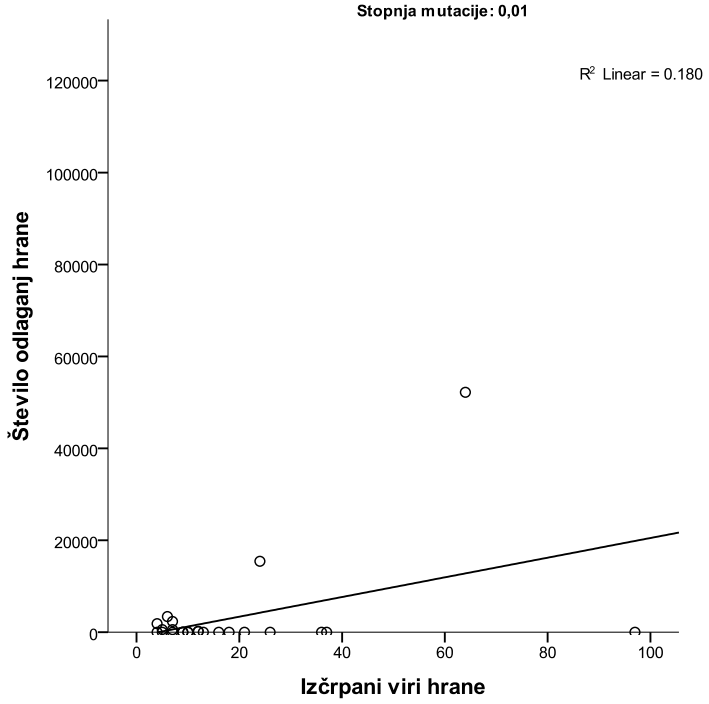


Slika 4.7: Prikaz odnosa med skupnim številom odlaganj hrane in izčrpanjem virov hrane v času za eno izmed kolonij s stopnjo mutacije 0,06.

Stopnja mutacije	Pearsonov r	r^2 (delež pojasnjene variance)
0,01	0,424*	0,180
0,02	0,848**	0,719
0,03	0,902**	0,814
0,04	0,908**	0,824
0,05	0,929**	0,863
0,06	0,930**	0,865
0,07	0,806**	0,650
0,08	0,803**	0,645
0,09	0,586**	0,343
0,10	0,796**	0,634

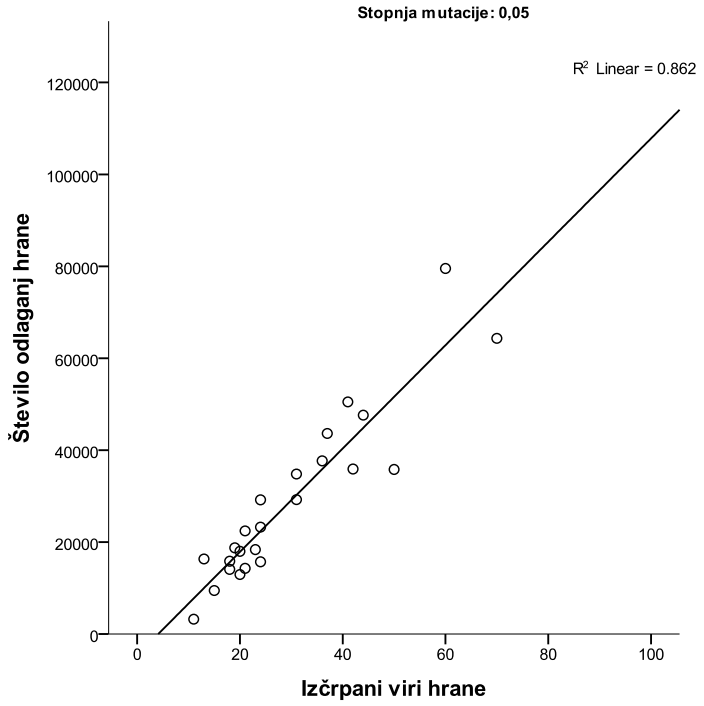
Tabela 4.2: Pearsonovi koeficienti korelacije med skupnim številom odlaganj hrane in skupnim številom izčrpanih virov hrane glede na stopnjo mutacije (*$p<0,05$; **$p<0,01$).

Na slikah 4.8 in 4.9 sta prikazana primera odnosa za nižjo in višjo stopnjo mutacije. Vidimo lahko, da je pri stopnji mutacije 0,01 veliko kolonij, ki so uspele izčrpati precej virov hrane, vendar pa hrane niso odlagale doma (Slika 4.8). Nasprotno pa se odnos med številom izčrpanih virov hrane in številom odlaganj hrane pri stopnjah mutacije med 0,03 in 0,06 skorajda prilega regresijski premici, kar je razvidno tudi iz primera za stopnjo mutacije 0,05 na sliki 4.9.



Slika 4.8: Prikaz odnosa med skupnim številom izčrpanih virov hrane in skupnim številom odlaganj hrane za kolonije s stopnjo mutacije 0,01.

Agenti so za prvo odlaganje hrane v povprečju porabili 22,52 minute (če upoštevamo le kolonije, ki so uspele odložiti hrano vsaj enkrat). Kot je razvidno iz Tabele 4.3, se pri tem kolonije razlikujejo tudi glede na stopnjo mutacije, pri čemer so največ časa porabile kolonije z nizkimi stopnjami mutacije (0,01 in 0,02), najmanj časa pa kolonije z višjimi stopnjami mutacije (0,06 - 0,10). Podobno velja tudi za razpršenost.



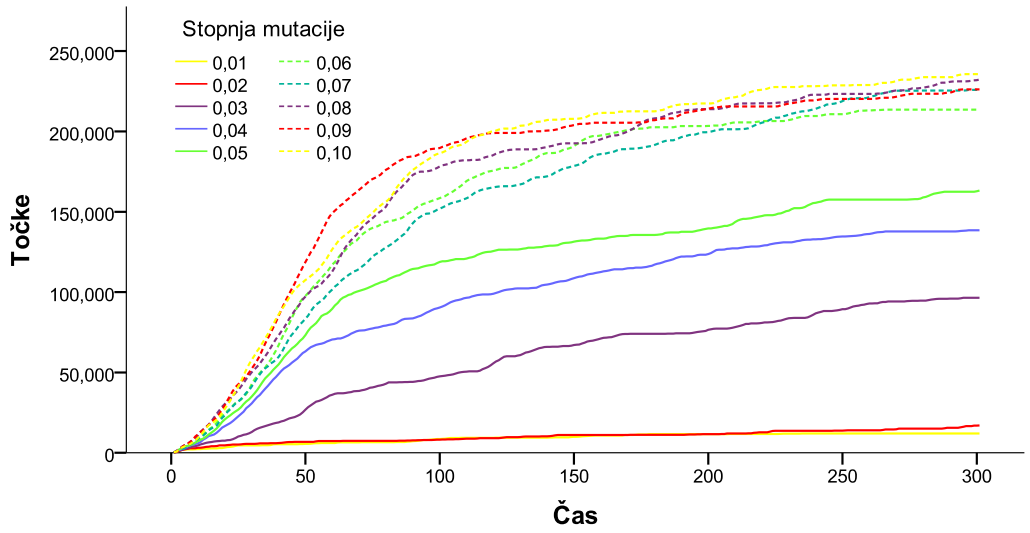
Slika 4.9: Prikaz odnosa med skupnim številom izčrpanih virov hrane in skupnim številom odlaganj hrane za kolonije s stopnjo mutacije 0,05.

Stopnja mutacije	N	Minimum	Maksimum	M	SD
0,01	13	5	228	69,15	74,465
0,02	17	3	299	77,47	90,145
0,03	24	3	236	44,25	58,602
0,04	24	1	47	15,25	13,079
0,05	24	3	83	17,29	24,676
0,06	24	1	24	7,92	5,664
0,07	24	4	26	10,25	5,550
0,08	24	2	19	7,46	4,222
0,09	24	3	16	6,71	2,896
0,10	24	3	18	6,83	3,679

Tabela 4.3: Povprečni čas, porabljen za prvo odlaganje hrane glede na stopnjo mutacije za kolonije, ki so uspele hrano odložiti vsaj enkrat.

4.3 Število točk najboljših agentov

Glede na to, da se agenti tekom obstoja kolonije razvijajo in napredujejo, lahko pričakujemo, da bodo točke najboljših agentov s časom rastle. Prav tako bi lahko pričakovali, da bo število točk najboljših agentov (znotraj obstoja posamezne kolonije) odvisno od stopnje mutacije, saj pri nizkih stopnjah mutacije ne prihaja do tako velikih razlik med agenti in tako ne nastajajo zelo uspešni agenti. Ravno obratno pa lahko pričakujemo za kolonije z visoko stopnjo mutacije. Pridobljeni rezultati so naša pričakovanja potrdili (Slika 4.10).



Slika 4.10: Prikaz števila točk najboljših agentov v koloniji v času glede na stopnjo mutacije.

5. Možnosti za izboljšave

Po izrisovanju grafičnih elementov se je za ozko grlo izkazalo predvsem iskanje feromonskih sledi. Sicer smo to funkcijo že precej izboljšali tekom razvoja, vendar obstajajo dodatne možnosti za optimizacijo.

Ena izmed možnih izboljšav je vezana na časovno zahtevno agentovo pregledovanje razpršenih tabel, ki vsebujejo podatke o feromonskih sledeh. Trenutno agent pregleda štiri med seboj prekrivajoče se razpršene tabele, ki vsebujejo podatke o feromonskih sledeh v njegovi okolici, in v vsaki poišče najmočnejšo feromonsko sled (maksimalno vrednost) ter se glede na rezultate odloči za nadaljnjo smer gibanja.

Možna bi bila uvedba dodatne tabele z velikostjo, enako številu trenutnih razpršenih tabel s feromoni. Ta bi ob vsakem osveževanju razpršenih tabel shranila podatke o velikosti in poziciji najmočnejših feromonskih sledi za vsako razpršeno tabelo posebej. Tako bi lahko poiskali najmočnejšo feromonsko sled s preprosto primerjavo štirih vrednosti.

5.1 Možnosti popravkov in razvoja simulacije

Nadgradnja simulacije je vezana na bolj realistično in nevarno okolje, na specializacijo in spremembo delovanja agentov ter na razvoj kolonije.

5.1.1 Spremembe na ravni okolja

Okolje bi lahko naredili bolj dinamično – postavili bi lahko nevarne objekte, ovire, vodna telesa, ki bi agentom oteževala gibanje, nekatera pa bi jih celo ogrožala. Dodali bi lahko različne tipe virov hrane glede na velikost (manjši, večji), statičnost (stacionarni, premični viri hrane), dostopnost (npr. vir, do katerega je potrebno priti okoli ovir) in težavnost pridobivanja (npr. vir, za katerega mora sodelovati večje število agentov in za katerega se je treba boriti, kot se npr. morajo za žuželke boriti mravlje). Kolonije bi se lahko specializirale za nabiranje le določenega tipa hrane ali pa bi pobirale vse tipe hrane. V simulaciji bi lahko bilo hkrati tudi več kolonij, ki bi med seboj tekmovale za hrano.

5.1.2 Spremembe na ravni agenta

Na ravni agenta bi lahko dodali nove načine razvoja oz. učenja, spremenili delovanje nevronske mreže in uvedli specializacijo agentov.

V obstoječi simulaciji je pri agentih prisotno le genotipsko učenje – agent tekom svojega življenja ne more izboljšati svojega delovanja, zato potekata dejansko učenje in razvoj le na nivoju kolonije. Možna izboljšava bi bila uvedba fenotipskega učenja, ki omogoča spremembe na ravni agenta, in sicer s pomočjo povratnega razširjanja (*back-propagation*). Tako bi se lahko manj uspešni agenti učili od bolj uspešnih agentov na podlagi vnaprej določenega nabora učnih primerov.

Nevronska mreža, ki predstavlja agentove možgane, ima v obstoječi simulaciji le pozitivne uteži, kar omogoča le vzpodbujevalne oz. ekscitacijske povezave. Smiselno bi bilo implementirati in opazovati delovanje agentov, če bi jim omogočili, da razvijejo tudi povezave z negativnimi utežmi, ki bi delovale zaviralno (inhibitorno; posamezen vhod bi tako lahko zmanjševal možnost

pojavljanja določenega vedenja).

Znotraj kolonije bi lahko na podlagi poznavanja organiziranosti in delovanja mravljišč poskušali uvesti različne tipe agentov, ki bi imeli različne vloge (npr. delavec, bojevnik, nabiralec) in fizične lastnosti (hitrost, velikost, velikost in kompleksnost možganov), ki bi jih, tako kot trenutno obnašanje, razvili s pomočjo evolucije. Tovrstni pristop bi zahteval bolj kompleksne nevronske mreže z dodatnimi vhodi (npr. prisotnost nasprotnikov) in izhodi (npr. bojevanje, gradnja) ter bolj kompleksen sistem ocenjevanja agentov.

Če bi kolonijam omogočili specializacijo, bi bilo zanimivo videti, ali bi bili agenti enaki in bi srednje dobro opravljali vse naloge ali pa bi se specializirali za opravljanje različnih nalog.

5.1.3 Spremembe na ravni kolonije

V trenutni simulaciji pridobiva kolonija nove agente ne glede na uspešnost. Da bi s pomočjo simulacije razvili ne le optimalno delujočega agenta, ampak tudi optimalno delujočo kolonijo, bi lahko kolonijam omogočili, da se razvijajo organsko, ne da bi bila njihova velikost konstantna in vnaprej določena. Tako bi kolonija rastla glede na količino nabrane hrane, saj bi za vsakega novega agenta potrebovala hrano. Če kolonija ne bi bila uspešna, bi zaradi pomanjkanja hrane in s tem novih agentov, izumrla.

Simulacijo lahko primerjamo s predstavljenimi tremi tipi vedenja mravelj (poglavje 1.4). Agenti sicer ne skušajo privabiti drugih k novo najdenemu viru hrane, vendar puščajo za seboj feromonske sledi. Tako sicer ne izvajajo skupinskega rekrutiranja, vendar pa feromonske sledi omogočajo masovno rekrutiranje, za katerega je značilno, da je feromonska sled dovolj ojačana, da ji agenti sledijo brez vodenja. Da bi agenti poiskali tudi druge vire hrane, jim omogočamo naključno raziskovanje. Pri nekaterih vrstah mravelj je tako vedenje obratno sorazmerno z močjo feromonske sledi in premo sorazmerno

z oddaljenostjo od gnezda. Da bi agentom omogočili tudi razvoj tovrstnega vedenja, bi jim kot dodaten vhod lahko dodali oddaljenost od doma.

6. Sklepi in ugotovitve

Izdelali smo simulacijo umetnega življenja za kolonijo agentov. V simulaciji smo implementirali okolje, v katerem se nahajajo viri hrane in dom kolonije, manjše enote agentov, ki imajo svoje možgane, ter skupaj tvorijo večjo enoto, imenovano kolonija. Cilj vsake kolonije je razviti agente, ki bodo skrbeli tako zase (nabirali in jedli hrano) kot tudi delovali v dobro kolonije (odlagali hrano doma). Da bi omogočili razvoj agentov na nivoju kolonije, smo uporabili kombinacijo nevronskih mrež in genetskih algoritmov. Vsak agent ima možgane, ki so predstavljeni z nevronsko mrežo in je (s precej unikatno kombinacijo uteži na povezavah znotraj nevronske mreže) predstavnik ene izmed možnih poskusnih rešitev razvoja optimalno delujočega agenta. S pomočjo genetskih algoritmov smo z dedovanjem in mutacijo že obstoječih nevronskih mrež zagotovili nastajanje novih poskusnih rešitev ter s tem razvoj kolonije. Komunikacijo med agenti smo implementirali s pomočjo puščanja feromonskih sledi, kar predstavlja zelo poenostavljeno obliko komunikacije med mravljami.

Po končani izdelavi simulacije nas je zanimalo, kako uspešne so posamezne kolonije glede na stopnjo mutacije pri dedovanju in kakšni so njihovi tipični vzorci vedenja. Ugotovili smo, da se kolonije glede na stopnjo mutacije precej razlikujejo. Kolonije z nizkimi stopnjami mutacije so namreč v povprečju izčrpale manjše število virov hrane, porabile več časa, da so izčrpale prvi vir hrane, v manjši meri nabrano hrano odlagale doma (oz. je v nekaterih

primerih niso nikoli odložile) ter imele v povprečju nižje število točk pri najbolj uspešnih agentih. Na drugi strani pa so imele takšne kolonije tudi najvišje zabeleženo število izčrpanih virov hrane. Takšni rezultati so za nizke stopnje mutacije smiselni, saj se zaradi manjših razlik med agenti zelo težko in počasi prilagajajo novim problemom, hkrati pa v primeru, da se razvije dober agent, tega pogosto kopirajo. Zaradi počasnega prilagajanja na probleme bi bilo zanimivo raziskati, kako se kolonije z nizko stopnjo mutacije prilagodijo, če jim damo za razvoj na voljo dosti več časa.

Glede na količino pobrane hrane oz. izčrpanih virov so se kot najbolj optimalne izkazale stopnje mutacije od 0,03 do 0,05, vendar pa so najhitreje našle hrano (oz. izčrpale prvi vir hrane) kolonije s stopnjami mutacije med 0,05 in 0,10. Najbolj uspešno so hrano shranjevale kolonije s stopnjo mutacije med 0,05 in 0,08, pri čemer pa so imele najvišjo korelacijo med skupnim številom odlaganj hrane in skupnim številom izčrpanih virov hrane kolonije s stopnjo mutacije od 0,03 do 0,06. Glede na število točk najboljših agentov se uvrščajo visoko vse kolonije s stopnjami mutacije med 0,05 in 0,10. Vidimo lahko, da so kolonije z nekaterimi stopnjami mutacije bolj uspešne pri iskanju, nekatere pri nabiranju, nekatere pa pri shranjevanju hrane. Zdi se, da je stopnja mutacije 0,05 tista, ki je v vseh navedenih primerih uvrščena zelo visoko, kar je najverjetneje odraz dobrega ravnovesja med zmožnostjo razvoja novih agentov, ko kolonija še ni uspešna, in zmožnostjo hitrega kopiranja uspešnih agentov. Zanimivo bi bilo raziskati, kako bi se vedle kolonije s še višjimi stopnjami mutacije in ali bi tudi pri vedenjih, pri katerih so bile uspešne kolonije s stopnjo mutacije 0,10, prišlo zaradi prevelike raznolikosti agentov do upada uspešnosti.

Zaradi primerjave uspešnosti kolonij z različnimi stopnjami mutacij smo za vse kolonije nastavili enotne parametre, vendar pa simulacija omogoča spreminjanje skoraj vseh parametrov preko menija. Tako lahko v simulaciji preprosto dodamo večje število kolonij, spreminjamo število agentov in njihovih lastnosti, čas obstoja kolonije in postavljanje virov hrane. Simulacija je

osnova za nadaljnji razvoj, možnosti za nadgradnje in dodajanje parametrov pa je veliko. Nadaljnji razvoj bi se lahko gibal k večji optimizaciji že obstoječega sistema, dodajanju kompleksnih vedenj agentov in k spreminjanju okolja, v katerem agenti delujejo. Takšne simulacije umetnega življenja skozi modeliranje evolucije komunikacije in sodelovanja posameznikov pri nižjih življenjskih oblikah omogočajo vsaj malo vpogleda v evolucijo in naravo človeške inteligentnosti.

Literatura

[1] M. G. Dyer, "Toward synthesizing artificial neural networks that exhibit cooperative intelligent behavior: Some open issues in artificial life," *Artificial Life*, vol. 1, no. 1_2, pp. 111–134, 1993.

[2] L. Steels, "The artificial life roots of artificial intelligence," *Artificial life*, vol. 1, no. 1_2, pp. 75–110, 1993.

[3] J. D. S. D. W. Larry, J. Eshelman, and B. M. F. C. B. Manor, "Combinations of genetic algorithms and neural networks: A survey of the state of the art," in *COGANN-92: International Workshop on Combinations of Genetic Algorithms and Neural Networks, June 6, 1992, Baltimore, Maryland*. IEEE Computer Society, 1992, p. 0.

[4] J. Branke, "Evolutionary algorithms for neural network design and training," in *In Proceedings of the First Nordic Workshop on Genetic Algorithms and its Applications*. Citeseer, 1995.

[5] D. J. Montana and L. Davis, "Training feedforward neural networks using genetic algorithms." in *IJCAI*, vol. 89, 1989, pp. 762–767.

[6] B. Kröse, B. Krose, P. van der Smagt, and P. Smagt, "An introduction to neural networks," 1993.

[7] A. Blum, *Neural Networks in C++: An Object-oriented Framework for Building Connectionist Systems.* New York, NY, USA: John Wiley & Sons, Inc., 1992.

[8] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 26, no. 1, pp. 29–41, 1996.

[9] D. Angus, "Ant colony optimisation: From biological inspiration to an algorithmic framework," Technical Report No. TR013, Tech. Rep., 2006.

[10] (2015, junij) Artificial life — Wikipedia, the free encyclopedia. Dostopno na: https://en.wikipedia.org/?title=Artificial_life

[11] A. Uršič, "Evolucija nevronskih mrež," Diplomsko delo, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, Ljubljana, 2012.

[12] (2015, junij) Artificial neural network — Wikipedia, the free encyclopedia. Dostopno na: https://en.wikipedia.org/wiki/Artificial_neural_network